由浅入深学 Java

——基础、进阶与必做260题

◎李志刚　等编著

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

内 容 简 介

本书讲解了 Java 方方面面的知识，全书共分为 4 篇共 28 章，第 1 篇从最基本的 JDK 安装讲起，包括基本语法与数据类型、数组、字符串、程序控制语句、面向对象编程、继承、多态、接口等内容；第 2 篇讲解了图形界面开发基础、Swing 编程、事件处理、数字处理、异常处理、文件的输入与输出、反射、线程、正则表达式、封装类、容器、泛型、数据结构、数据库编程和 Applet 等高级技术；第 3 篇讲解了 Socket、JSP、Servlet 等网络编程技术；第 4 篇介绍了一个综合案例，让读者有提高的机会。

本书是一本 Java 编程自学手册，读者可以跟随书的讲解，边学习边上机，最好在学完一章后，独立做出每章的习题，这样可以带着问题学习，进步得会更快。

未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。
版权所有，侵权必究。

图书在版编目（CIP）数据

由浅入深学 Java：基础、进阶与必做 260 题 / 李志刚等编著. —北京：电子工业出版社，2011.7
（由浅入深学）
ISBN 978-7-121-13238-4

Ⅰ. ①由… Ⅱ. ①李… Ⅲ. ①JAVA 语言－程序设计－习题集 Ⅳ. ①TP312-44

中国版本图书馆 CIP 数据核字（2011）第 056070 号

策划编辑：胡辛征
责任编辑：贾 莉 李利健
印 刷：北京东光印刷厂
装 订：三河市皇庄路通装订厂
出版发行：电子工业出版社
北京市海淀区万寿路 173 信箱 邮编 100036
开 本：787×1092 1/16 印张：36 字数：968 千字
印 次：2011 年 7 月第 1 次印刷
印 数：4000 册 定价：69.80 元（含 DVD 光盘 1 张）

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

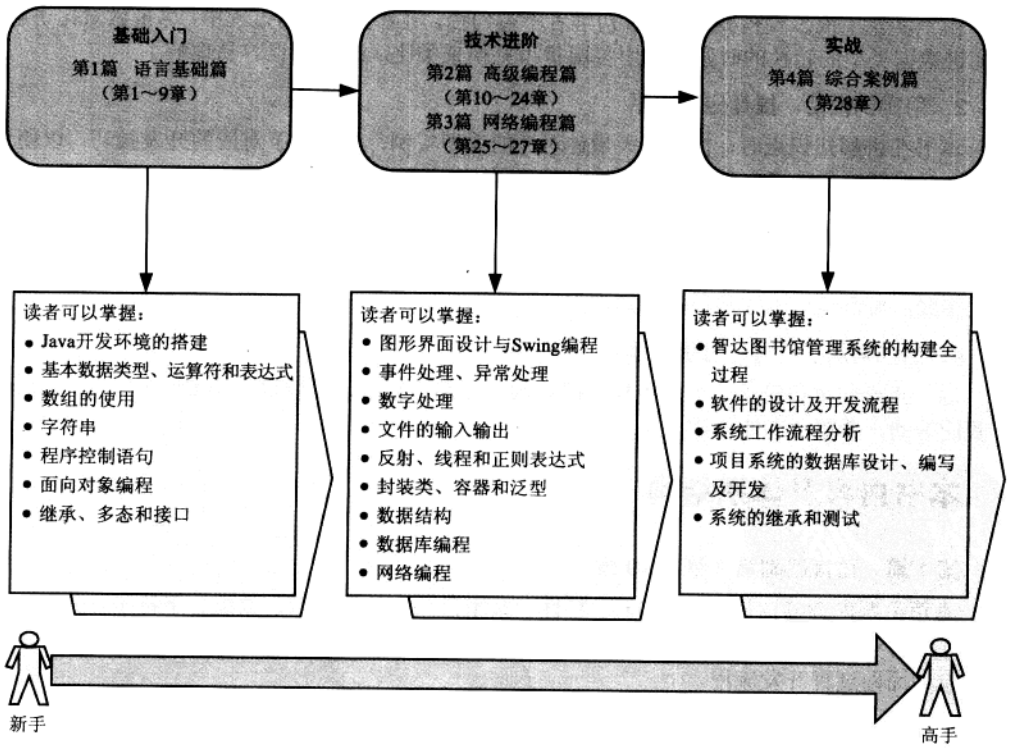
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

前　言

Java 是当前应用最广泛的语言，根据 TIOBE 开发语言排行榜显示，它始终位于第一的位置。在各大求职招聘网站上也可以看到，关于 Java 开发的职位非常多，因此掌握了该语言，就能轻松获得一个不错的职位。

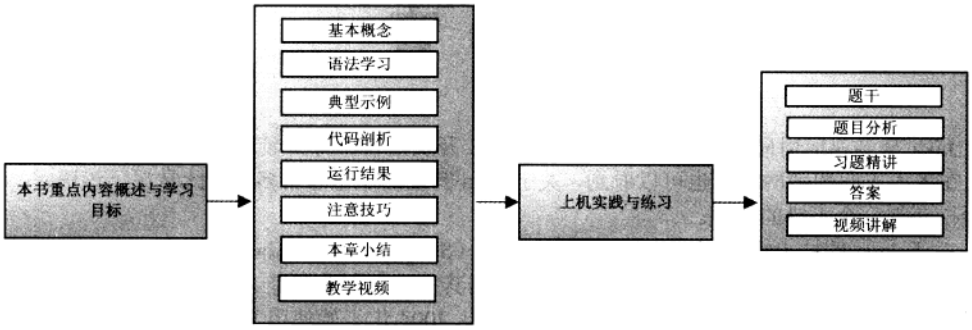
关于 Java 的技术很多，比如流行的 SSH 组合（Struts+Spring+Hibernate）、EJB、J2ME、JSF、JPA，等等，经常闹得人一头雾水，其实万变不离其宗，只要掌握好 Java 基础，在学习这些技术时都可以手到擒来。笔者有 8 年的开发经验，从 2004 年开始使用 Java 进行开发，在使用过程中颇有体会，为了帮助新手快速掌握 Java，特编写了本书。

本书内容体系与学习梯度



Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

本书内容编写体例



本书特色

1．采用比喻的手法进行讲解

由于涉及了很多编程概念，很多初学者不得门径，笔者在写作的过程中，尽量做对比讲解，比如讲数据类型的转换的时候，就用各国货币兑换来类比，这样就很容易理解了。

2．用实例说话，提高应用水平

本书在讲解知识点时，贯穿了大量针对性的典型实例，并给出了对应的开发技巧，以便让读者更好地理解各种概念和方法，体验实际编程，加深理解程度

3．每章配有练习题，帮助读者自检

本书的练习题丰富，且都是具有可操作性的习题，读者在阅读完章节知识点后，通过这些编程实践，可迅速巩固并掌握基础知识点。

4．给代码加了行号，方便阅读

这一点看起来很简单，可真正操作的时候一点都不轻松，可以说每个 01、02、03 都凝结了我的劳动，希望读者能喜欢。

本书内容及体系结构

第 1 篇　语言基础篇（第 1～9 章）

本篇主要内容包括初步认识 Java 特性、基本语法和数据类型、数组、字符串、程序控制语句、面向对象编程、继承、多态、接口等内容。通过本篇的学习，读者可以掌握 Java 的基础知识点和面向过程开发流程。

第 2 篇　高级编程篇（第 10～24 章）

本篇主要内容包括：图形界面开发基础、Swing 编程、事件处理、数字处理、异常处理、文件的输入和输出、反射、线程、正则表达式、封装类、容器、泛型、数据结构、数据库编程和 Applet 等内容。这一篇囊括了 Java 开发的所有核心技术，各种编程思想也在这里最集中，读者要仔细学习。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第 3 篇　网络编程篇（第 25~27 章）

网络编程的内容很多，可以单独写一本很厚的书，在本篇中进行了简单的介绍，让读者大致有个入门的认识，主要讲解了 Socket 编程、JSP 和 Servlet 等内容。

第 4 篇　综合案例篇（第 28 章）

本篇只有一章，讲解了智达图书馆管理系统的开发过程，通过本篇，读者可以全面应用前面所学的开发技术进行软件项目开发，达到可以独立开发项目的水平。

实用超值的 DVD 光盘

为了帮助读者比较直观地学习，本书附赠 DVD 光盘，内容包括多媒体视频、电子教案（PPT）和实例源代码等。

- 多媒体教学视频

配有长达 12 个小时手把手教学视频，讲解关键知识点界面操作和书中的一些综合练习。

- 电子教案（PPT）

本书可以作为高校相关课程的教材或课外辅导书，所以，笔者特别为本书制作了电子教案（PPT），以便老师教学使用。

适合阅读本书的读者

- 没有任何语言学习经验的 Java 初学者。
- 有志于成为 Java 程序员的读者。
- 了解一些 C 语言，想要学习 Java 的读者。
- 上过培训班，感觉学得不够深入的读者。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

目　录

第 1 篇　语言基础篇

第 1 章　Java 入门（ 教学视频：18 分钟）……1

1.1　Java 简介……1
1.2　Java 的特点……1
1.3　JDK 的安装及配置……2
1.3.1　JDK 的下载安装……2
1.3.2　配置环境变量……3
1.3.3　JDK 环境的测试……3
1.4　Java 语言开发工具……4
1.4.1　Eclipse 的安装与使用……4
1.4.2　MyEclipse 的安装……6
1.4.3　使用开发工具编写“HelloWorld!”……7
1.5　习题……9

第 2 章　Java 基本数据类型（ 教学视频：46 分钟）……10

2.1　基本数据类型……10
2.1.1　布尔类型……10
2.1.2　字符型……11
2.1.3　整型……11
2.1.4　实型……12
2.2　数据类型的转换……12
2.2.1　隐式数据类型转换……13
2.2.2　显式类型转换……14
2.3　运算符和表达式……14
2.3.1　算术运算符及表达式……15
2.3.2　关系运算符及表达式……16
2.3.3　逻辑运算符及表达式……17
2.3.4　位运算符及表达式……18
2.3.5　赋值运算符及表达式……19
2.4　习题……19

第 3 章　数组（ 教学视频：31 分钟）……23

3.1　数组基础……23
3.1.1　创建数组……23
3.1.2　数组的初始化……24
3.1.3　数组的使用……25
3.2　数组的排序……27

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

3.2.1 选择排序法 ····27
3.2.2 冒泡排序法 ····28
3.3 多维数组 ····28
3.4 习题 ····30
第 4 章 字符串（ 教学视频：27 分钟） ····34
4.1 字符串 ····34
4.1.1 声明字符串 ····34
4.1.2 创建字符串 ····34
4.2 字符串操作 ····35
4.2.1 字符串的比较 ····35
4.2.2 连接字符串 ····36
4.2.3 复制字符串 ····37
4.2.4 更改字符串的内容 ····37
4.2.5 获取字符串的长度 ····38
4.2.6 分割字符串 ····39
4.3 StringBuffer ····39
4.4 习题 ····40
第 5 章 程序控制语句（ 教学视频：24 分钟） ····43
5.1 顺序结构 ····43
5.2 条件语句 ····43
5.2.1 if 语句 ····44
5.2.2 switch 语句 ····47
5.3 循环语句 ····48
5.3.1 for 循环 ····48
5.3.2 while 循环 ····50
5.3.3 do-while ····51
5.4 中断控制 ····52
5.4.1 continue ····52
5.4.2 break ····53
5.4.3 return ····53
5.5 习题 ····54
第 6 章 面向对象程序设计（ 教学视频：10 分钟） ····59
6.1 面向对象程序设计基础 ····59
6.2 类 ····59
6.2.1 定义及使用类 ····59
6.2.2 主方法 ····61
6.2.3 构造方法 ····61
6.2.4 成员变量 ····64
6.2.5 局部变量 ····66
6.2.6 this 关键字 ····67
6.3 对象 ····69

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

6.3.1 创建和使用对象 ····· 69
6.3.2 对象行为和属性的访问 ····· 71
6.3.3 比较对象 ····· 72
6.3.4 销毁对象 ····· 73
6.4 习题 ····· 74
第 7 章 继承（教学视频：13 分钟） ····· 78
7.1 继承概述 ····· 78
7.1.1 什么是继承 ····· 78
7.1.2 实现继承 ····· 78
7.2 继承关系 ····· 79
7.2.1 单一继承关系 ····· 79
7.2.2 多重继承关系 ····· 80
7.3 抽象类 ····· 81
7.4 使用关键字 final ····· 82
7.4.1 常量 ····· 82
7.4.2 方法 ····· 83
7.4.3 类 ····· 83
7.5 Object 类 ····· 84
7.5.1 Object 对象 ····· 84
7.5.2 重写 equals 方法和 toString 方法 ····· 84
7.6 习题 ····· 86
第 8 章 多态（教学视频：15 分钟） ····· 90
8.1 多态的概述 ····· 90
8.1.1 重写 ····· 90
8.1.2 重载 ····· 92
8.2 绑定 ····· 94
8.2.1 静态绑定 ····· 94
8.2.2 动态绑定 ····· 95
8.3 习题 ····· 96
第 9 章 接口（教学视频：20 分钟） ····· 100
9.1 认识接口 ····· 100
9.1.1 概述 ····· 100
9.1.2 定义接口 ····· 100
9.1.3 实现接口 ····· 101
9.1.4 多重实现接口 ····· 102
9.1.5 接口的引用 ····· 103
9.2 内部类 ····· 105
9.2.1 内部类概述 ····· 105
9.2.2 匿名内部类 ····· 106
9.2.3 继承内部类 ····· 108
9.2.4 静态内部类 ····· 109

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

9.2.5 引用外部类的对象 ……109
9.3 包 ……111
9.3.1 创建包 ……111
9.3.2 导入包 ……112
9.4 习题 ……112

第 2 篇 高级编程篇

第 10 章 图形界面开发基础（ 教学视频：43 分钟） ……119

10.1 AWT 简介 ……119
10.2 容器 ……119
10.2.1 窗口 ……119
10.2.2 面板 ……121
10.3 组件 ……122
10.3.1 按钮和标签 ……122
10.3.2 复选框 ……124
10.3.3 文本域 ……125
10.3.4 列表 ……127
10.4 布局管理器 ……129
10.4.1 网格布局管理器 ……129
10.4.2 顺序布局管理器 ……132
10.4.3 边界布局管理器 ……134
10.4.4 卡片布局管理器 ……135
10.5 习题 ……137

第 11 章 Swing（ 教学视频：42 分钟） ……145

11.1 Swing 基础 ……145
11.2 Swing 组件 ……145
11.2.1 按钮 ……146
11.2.2 复选框 ……146
11.2.3 单选按钮 ……147
11.2.4 组合框 ……149
11.2.5 进度条 ……150
11.2.6 表格 ……150
11.2.7 树 ……150
11.2.8 文本框 ……151
11.3 Swing 常用窗体 ……153
11.3.1 JFrame 窗体 ……153
11.3.2 JDialog 窗体 ……154
11.4 Swing 常用面板 ……156
11.5 习题 ……158

第 12 章 事件处理（ 教学视频：33 分钟） ……167

12.1 事件类 ……167

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

12.1.1 事件类分类 ····· 167
12.1.2 动作事件类 ····· 168
12.1.3 调整事件类 ····· 168
12.1.4 焦点事件类 ····· 168
12.1.5 项目事件类 ····· 169
12.1.6 键盘事件类 ····· 169
12.1.7 鼠标事件类 ····· 171
12.1.8 窗口事件类 ····· 172
12.2 事件监听器 ····· 173
12.2.1 事件监听器的接口实现 ····· 174
12.2.2 应用事件监听器 ····· 174
12.2.3 事件监听器的特点 ····· 175
12.3 事件适配器 ····· 175
12.4 事件应用方法 ····· 176
12.4.1 内部类方式应用 ····· 176
12.4.2 匿名类方式应用 ····· 177
12.4.3 外部类方式应用 ····· 178
12.5 习题 ····· 179
第 13 章 数字处理（ 教学视频：21 分钟） ····· 186
13.1 Math 类介绍 ····· 186
13.2 猜数字游戏 ····· 187
13.3 判断是否是数字 ····· 188
13.4 字符串转换成数字 ····· 190
13.5 格式化数字 ····· 191
13.6 习题 ····· 193
第 14 章 异常处理（ 教学视频：12 分钟） ····· 196
14.1 异常概述 ····· 196
14.1.1 异常发生的原因 ····· 197
14.1.2 异常处理机制 ····· 197
14.2 异常的结构 ····· 197
14.2.1 异常的层次结构 ····· 197
14.2.2 Java 中的运行时异常类 ····· 198
14.3 异常的处理 ····· 200
14.3.1 获取异常信息 ····· 200
14.3.2 异常声明 ····· 202
14.3.3 抛出异常 ····· 203
14.4 用户自定义异常类 ····· 206
14.4.1 创建自己的异常类 ····· 206
14.4.2 使用自己的异常类 ····· 207
14.5 习题 ····· 208

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第 15 章　文件的输入与输出（ 教学视频：30 分钟）……213
15.1　流的概念……213
15.2　输入输出流……213
15.3　字节流……214
15.3.1　字节输入、输出流……214
15.3.2　字节文件输入、输出流……215
15.3.3　字节缓冲输入、输出流……216
15.3.4　字节数据输入、输出流……218
15.3.5　字节对象输入、输出流……220
15.4　字符流……222
15.4.1　字符读、写流……222
15.4.2　字符输入、输出流……223
15.4.3　文件读、写字符流……224
15.4.4　字符缓冲读、写流……226
15.5　文件类……226
15.5.1　File 类……226
15.5.2　File 类应用……227
15.5.3　RandomAccessFile 类……228
15.5.4　RandomAccessFile 应用……229
15.6　习题……232
第 16 章　反射（ 教学视频：20 分钟）……240
16.1　反射概述……240
16.1.1　使用 Class 类……242
16.1.2　成员变量……243
16.1.3　成员方法……243
16.1.4　创建实例……244
16.2　应用反射……244
16.2.1　访问类变量……244
16.2.2　调用类方法……246
16.3　反射的安全性……247
16.4　习题……249
第 17 章　线程（ 教学视频：35 分钟）……255
17.1　线程的概念……255
17.2　线程的创建……255
17.2.1　使用 Thread 创建线程……255
17.2.2　使用 Runnable 创建线程……257
17.2.3　两种方法比较……258
17.2.4　创建多线程……258
17.3　线程的使用……260
17.3.1　开启线程……260
17.3.2　线程休眠……261

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

17.3.3 等待线程 ····263
17.4 线程的控制 ····265
17.4.1 线程的优先级 ····265
17.4.2 线程让步 ····267
17.4.3 线程同步 ····269
17.4.4 多线程的死锁问题 ····273
17.5 线程池 ····274
17.6 Swing 线程应用 ····277
17.6.1 多线程模拟钟表 ····277
17.6.2 线程的优先级模拟赛车 ····281
17.6.3 使用断点实现计数器 ····283
17.6.4 使用线程实现屏保“璀璨星空” ····286
17.6.5 实现下雪场景 ····288
17.6.6 实现瀑布效果 ····290
17.6.7 滚动字幕 ····292
17.7 习题 ····294
第 18 章 正则表达式（ 教学视频：17 分钟） ····302
18.1 正则表达式语法 ····302
18.2 正则表达式的使用 ····304
18.2.1 Pattern 类 ····304
18.2.2 Matcher 类 ····305
18.2.3 综合运用 ····306
18.3 String 类中的正则表达式应用 ····307
18.3.1 模式匹配检查 ····307
18.3.2 查找替换 ····308
18.3.3 字符串分析 ····309
18.4 习题 ····311
第 19 章 封装类（ 教学视频：21 分钟） ····314
19.1 常用的数据类型包装类 ····314
19.1.1 Integer 类 ····314
19.1.2 Double 类 ····315
19.1.3 Number 类 ····317
19.1.4 Float 类 ····317
19.1.5 Boolean 类 ····318
19.1.6 Byte 类 ····320
19.2 StringBuffer 类 ····321
19.2.1 StringBuffer 简介 ····321
19.2.2 StringBuffer 类的方法 ····321
19.3 日期操作类 ····323
19.3.1 Date 类 ····324
19.3.2 DateFormat 类 ····325
19.4 习题 ····326

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第 20 章　容器（ 教学视频：23 分钟）……331
20.1　容器概述……331
20.2　迭代器……331
20.3　List……333
20.3.1　List 接口方法……333
20.3.2　链表迭代器接口方法……333
20.3.3　链表和数组链表……334
20.3.4　List 应用……334
20.4　集 Set……339
20.4.1　Set 接口的方法……339
20.4.2　哈希集和树集……340
20.4.3　实现 Set……341
20.5　映射 Map……343
20.5.1　Map 接口方法……343
20.5.2　哈希映射和树映射……344
20.5.3　哈希映射应用……345
20.5.4　实现 Map……346
20.6　习题……347
第 21 章　泛型（ 教学视频：15 分钟）……355
21.1　泛型概述……355
21.2　泛型格式……355
21.2.1　定义泛型类……355
21.2.2　类型变量的限制……358
21.3　泛型参数的继承和通配符……359
21.3.1　泛型的继承……359
21.3.2　泛型中的通配符……359
21.4　习题……360
第 22 章　数据结构（ 教学视频：39 分钟）……364
22.1　数据结构的基本概念……364
22.2　算法……364
22.2.1　算法概述……364
22.2.2　算法分析……364
22.2.3　递归算法……365
22.2.4　排序算法……366
22.3　链表……368
22.3.1　Java 中的链表……368
22.3.2　链表的添加……368
22.3.3　链表的删除……370
22.4　栈和队列……372
22.4.1　栈……372
22.4.2　队列……372

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

22.4.3 Java 实现栈 ····372
22.4.4 Java 实现队列 ····373
22.5 习题 ····374
第 23 章 数据库（ 教学视频：20 分钟） ····381
23.1 数据库的介绍 ····381
23.2 MySQL 概述 ····381
23.2.1 MySQL 简介 ····381
23.2.2 下载并安装 MySQL ····382
23.2.3 MySQL 使用 ····385
23.3 SQL 语句 ····387
23.3.1 查询语句 SELECT ····388
23.3.2 插入语句 INSERT ····388
23.3.3 更新语句 UPDATE ····388
23.3.4 删除语句 DELETE ····389
23.3.5 建表语句 CREATE ····389
23.3.6 修改表语句 ALTER ····389
23.3.7 删除表语句 DROP ····390
23.4 JDBC 设计 ····390
23.4.1 JDBC 常见接口和类 ····390
23.4.2 JDBC 数据库连接步骤 ····391
23.4.3 加载 JDBC 驱动程序 ····392
23.4.4 获取 Connection 连接对象 ····393
23.5 Java 与各种数据库连接 ····394
23.6 习题 ····395
第 24 章 Applet 编程（ 教学视频：15 分钟） ····399
24.1 Applet 类 ····399
24.1.1 Applet 简介 ····399
24.1.2 Applet 的开发流程 ····399
24.1.3 Applet 的方法 ····400
24.1.4 Applet 的生命周期 ····400
24.2 Applet 应用 ····402
24.2.1 Applet 中的文字处理 ····403
24.2.2 动态物体输出 ····408
24.2.3 图片处理 ····413
24.2.4 放大镜 ····415
24.3 习题 ····417

第 3 篇 网络编程篇

第 25 章 网络编程（ 教学视频：30 分钟） ····422
25.1 网络基本概念 ····422

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

25.2 TCP/IP ····422
25.2.1 TCP/IP 与网络协议 ····422
25.2.2 传输控制层（TCP） ····423
25.2.3 IP 应用 ····425
25.3 Socket ····429
25.3.1 Socket 类 ····429
25.3.2 ServerSocket 类 ····430
25.3.3 Socket 应用 ····431
25.4 URL ····435
25.4.1 URL 的使用 ····435
25.4.2 提交表单信息 ····437
25.4.3 简单的浏览器 ····437
25.5 习题 ····439
第 26 章 JSP（ 教学视频：31 分钟） ····445
26.1 JSP 运行环境搭建 ····445
26.1.1 Tomcat 下载与安装 ····445
26.1.2 Tomcat 的环境变量配置 ····446
26.2 基本语法 ····447
26.2.1 注释 ····447
26.2.2 声明 ····448
26.2.3 表达式 ····449
26.2.4 脚本集 ····449
26.3 JSP 指令元素 ····451
26.3.1 Include 指令 ····451
26.3.2 Page 指令 ····452
26.3.3 Taglib 指令 ····453
26.4 JSP 动作元素 ····454
26.4.1 <jsp:forward> ····454
26.4.2 <jsp:useBean> ····454
26.4.3 <jsp:setProperty> ····456
26.4.4 <jsp:getProperty> ····457
26.4.5 <jsp:include> ····458
26.4.6 <jsp:plugin> ····458
26.5 内置对象 ····459
26.5.1 out 对象 ····460
26.5.2 request 对象 ····460
26.5.3 response 对象 ····463
26.5.4 session 对象 ····466
26.5.5 application 对象 ····468
26.5.6 pageContext 对象 ····470
26.5.7 exception 对象 ····470
26.5.8 config 对象 ····471

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

26.6 习题 ····472
第 27 章 Servlet（ 教学视频：37 分钟） ····476
27.1 Servlet 简介 ····476
27.1.1 Servlet 接口 ····476
27.1.2 目录结构 ····476
27.1.3 编写 Servlet 程序 ····477
27.2 Servlet 的生命周期 ····480
27.3 HttpServlet 类 ····480
27.3.1 doGet ····480
27.3.2 doPost ····483
27.4 HttpServletResponse 类及状态响应码 ····488
27.5 Servlet 中的请求与响应 ····489
27.6 Servlet 之间的跳转 ····491
27.6.1 Forward ····491
27.6.2 Redirect ····493
27.6.3 Refresh ····495
27.7 JavaBean ····496
27.7.1 使用 JavaBean ····496
27.7.2 JSP 中的 JavaBean ····496
27.8 习题 ····499

第 4 篇 综合案例篇

第 28 章 智达图书馆管理系统 ····505
28.1 项目立项 ····505
28.1.1 需求分析 ····505
28.1.2 可行性分析 ····505
28.2 详细设计 ····506
28.2.1 系统目标 ····507
28.2.2 系统功能结构 ····507
28.2.3 构建开发环境 ····507
28.2.4 学生端系统预览 ····507
28.2.5 管理员端系统预览 ····509
28.3 数据库设计 ····510
28.3.1 数据库分析 ····511
28.3.2 E-R 图 ····511
28.3.3 数据表设计 ····512
28.3.4 导入数据 ····514
28.4 数据库接口 ····515
28.5 登录模块 ····516
28.5.1 登录界面 ····516
28.5.2 键盘易用性 ····517

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

28.5.3 管理员登录 ……518
28.5.4 学生登录 ……519
28.6 管理模块 ……520
28.6.1 管理端界面的初始化 ……520
28.6.2 管理端其他方法介绍 ……522
28.7 图书查询模块 ……524
28.7.1 查询界面 ……524
28.7.2 查询界面构造器 ……525
28.7.3 简单查询 ……526
28.7.4 高级查询 ……527
28.8 图书借阅预约模块 ……529
28.8.1 界面开发 ……530
28.8.2 显示图书信息 ……530
28.8.3 借阅预约功能 ……532
28.9 图书归还挂失模块 ……533
28.9.1 界面开发 ……534
28.9.2 查询借书记录 ……534
28.9.3 归还和挂失图书 ……535
28.9.4 挂失处理的开发 ……536
28.9.5 其他方法介绍 ……537
28.10 图书超期模块 ……539
28.10.1 界面开发 ……539
28.10.2 查询超期天数 ……540
28.10.3 查询及交纳欠款 ……541
28.11 学生管理模块 ……542
28.11.1 界面开发 ……542
28.11.2 响应管理事件 ……543
28.11.3 操作学生信息 ……543
28.12 图书管理模块 ……546
28.12.1 界面开发 ……546
28.12.2 响应管理事件 ……547
28.12.3 操作图书信息 ……547
28.12.4 修改图书信息 ……548
28.13 管理员管理模块 ……549
28.13.1 界面开发 ……549
28.13.2 响应管理事件 ……550
28.13.3 普通管理员功能 ……552
28.13.4 添加管理员 ……553
28.13.5 删除管理员功能 ……554
28.13.6 修改管理员信息 ……554
28.13.7 查询管理员信息 ……555
28.14 集成和测试 ……556
28.15 总结 ……557

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第 1 篇　语言基础篇

第 1 章　Java 入门

Java 语言是当前比较流行的语言，其使用范围非常广。本章首先介绍了 Java 的发展、特点及运行环境，然后介绍了 Java 开发工具的安装，并以一个“HelloWorld!”的示例来介绍开发工具的使用。



1.1　Java 简介

在计算机中有一些不同的语言，这些语言都有自己的语法和表达方式，能够以不同的表达方式让计算机来完成一些对应的功能，Java 语言就是其中的一种。

Java 的前身是一个由 Sun 公司为一些家用消费电子产品所开发的名为 Oak 的软件，但是这款软件并没有引起太大的关注，直到 1994 年，随着 Internet 的飞速发展，Sun 公司对 Java 的研究也进展迅速，使得一些非常著名的公司也注意到了 Java，并纷纷购买 Java 的使用权。

Java 也是一种网络设计语言，网络使得 Java 成为最为流行的编程语言，而 Java 也促进了网络的发展。Java 开发涉及的面很广，包括有桌面开发、网络开发和嵌入式开发等。

当前 IT 行业的 Java 技术人员还是比较短缺的，一般 Java 开发经验在一年以上的程序员，月薪基本在四千左右，而 Java 开发经验在三年以上的软件工程师，月薪会达到八千甚至更高，所以说，学习 Java 这门语言也是从事 IT 行业的人员很不错的选择。



1.2　Java 的特点

正如人与人是不同的，相较于其他的开发语言，Java 有一些自己的特点。

1．使用简单

在 Java 之前，有很多编程语言的基础往往很难，如 C 语言中的指针就是一个难点，有好多编程人员在工作了几年之后，还是搞不懂指针到底是怎么回事。这个问题被 Java 给解决了，在 Java 中已经去掉了难以理解的指针。

2．面向对象

现实生活中的所有实体事物都可以看做是对象。Java 的设计都是集中于对象及其接口的，并且提供了简单的类机制及动态的接口模型。

3．稳健性

Java 具有稳健性，在编译和运行时，都要对可能出现的一些问题进行检查，以消除错误的产生。

4．安全性

浏览网页时，可能会感染病毒或木马，这些木马或者病毒轻者能使用户电脑系统崩溃，重

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

者可以盗取用户的重要数据。针对这个问题，Java 也有解决方案，使用 Java 编写的 Applet 程序将 Java 小程序限制在一个“保险柜”中运行，程序不能读取本地数据，也无法向本地硬盘中写入数据。

5. 高性能

高效率的工作能够给公司或者工厂带来更多的利益。在软件开发中，让程序高效率地运行可以大大节省时间。Java 在刚刚出现的时候，运行性能常常为人所诟病，后来 Sun 公司意识到了这个问题，在以后的版本中不断提高 Java 的运行效率，现在，使用 Java 编写的程序不会比 C 语言慢。

6. 多线程

如今，农民收获小麦时，已经很少手工进行，大多数使用了联合收割机，在一片大的农田中，有时候会看到多台收割机同时作业，如果将每一台收割机都理解为一个线程的话，那多台同时作业就可以看做是多线程。

多线程也是 Java 的重要特点，多线程编程就是将程序任务分成几个并行的子任务，然后同时运行。例如，在同一个时刻总会有多个用户去访问同一个服务器，这时就要使用多线程来实现。

7. 跨平台性

英语虽然不是使用人数最多的语言，但它是世界上使用最广泛的语言，在世界各地，都有使用英语进行标注的路标，因此，一个会英语的人，走到哪里应该都不会迷路。Java 具有“一次编写，到处运行”的跨平台性，这是 Java 语言能够广泛流行的重要原因。使用 Java 语言编写的程序在不同的平台之间有非常好的可移植性。



1.3 JDK 的安装及配置

Java 程序的运行，需要底层的一些支持 Java 运行的基础运行程序，也就是 JDK（Java Software Development Kit）。普通的用户，在自己的计算机上运行 Java 程序之前，必须要安装 JDK。自 JDK 诞生以来，其版本在不断地升级，如 JDK1.4、JDK1.5、JDK1.6 等。

1.3.1 JDK 的下载安装

JDK 的安装文件可以从 Oracle 公司的官方网站下载，该网站还为不同操作系统提供了支持。这里仅提供在 Windows 操作系统下的安装步骤说明。

① 双击下载好的 JDK 安装文件，等待一段时间，会自动进入许可协议的对话框，单击“接受”按钮，接受许可协议，进入如图 1-1 所示的对话框。

② 如果需要自定义 JDK 的安装路径，单击“更改”按钮，选择 JDK 的安装路径，然后单击“下一步”按钮。如果不需要自定义 JDK 的安装路径，直接单击“下一步”按钮。

③ 等待一段时间后，会弹出如图 1-2 所示的对话框，这是设置 JRE（JRE 是 Java 的运行环境）安装路径的对话框。如果需要自定义 JRE 的安装路径，单击“更改”按钮，修改安装路径以后，单击“下一步”按钮。

④ 安装好以后，在弹出的对话框中单击“完成”按钮，这样就完成了 JDK 的安装。



说明：因为 JDK 是 Java 程序的开发环境，所以在 JDK 的安装文件中还包含了一个 JRE 文件，JRE 是 Java 的运行环境，在默认的情况下，JRE 和 JDK 一起安装。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

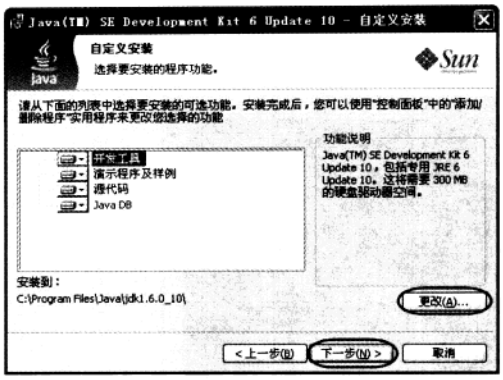


图 1-1　选择 JDK 安装路径

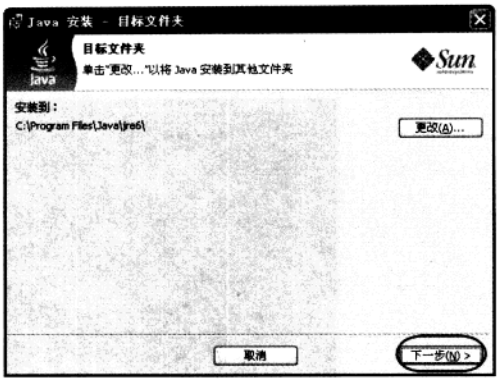


图 1-2　选择 JRE 安装路径

1.3.2　配置环境变量

正如公司里安装了一台服务器，安装完成后，需要告诉设备管理员，这台服务器放置的位置，以及如何使用。在计算机中配置操作系统的环境变量也是如此。在安装完 JDK 后，需要配置一下系统的环境变量，下面是配置环境变量的具体步骤。

① 右击“我的电脑”图标，选择“属性”菜单选项，弹出“系统属性”对话框，单击“高级”标签页，在出现的对话框中单击“环境变量”按钮，弹出“环境变量”对话框，单击“系统变量”栏的“新建”按钮，弹出“新建系统变量”对话框，在“变量名”文本框中输入“Java_home”，在“变量值”文本框中输入 JDK 的安装目录（如果是默认安装的，则输入 C:\Program Files\Java\jdk1.6.0_10，如图 1-3 所示），单击“确定”按钮完成配置。返回到“环境变量”对话框中，继续新建系统变量，在“变量名”文本框中输入“classpath”，在“变量值”文本框中输入“.”，单击“确定”按钮完成配置。

② 在“环境变量”对话框的“系统变量”列表中找到名为“Path”的变量，单击“Path”所在的行（如图 1-4 所示），再单击“编辑”按钮，弹出“编辑系统变量”对话框，在“变量值”文本框结尾处输入“;%Java_home%\bin”，单击“确定”按钮完成配置。



注意：一定要在“变量值”文本框中用“;”将多个变量值分开。

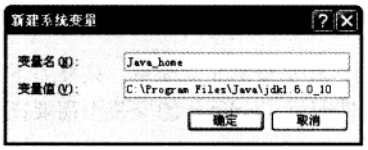


图 1-3　新建系统变量

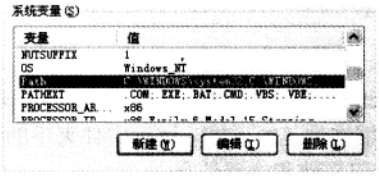


图 1-4　系统变量编辑

1.3.3　JDK 环境的测试

JDK 安装完成后，需要对其进行测试，查看安装或者环境变量是否都正常。

单击 Windows XP 桌面左下角的“开始”按钮，选择“运行”菜单选项，弹出“运行”对话框，在文本框中输入“cmd”，单击“确定”按钮，弹出命令提示符对话框，输入“java –version”命令。如果提示“java –version 不是内部或外部命令，也不是可运行的程序”，则说明环境变量

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

配置错误，需要进行检查，重新配置；如果显示中含有“1.6.0_10”字样，说明配置成功。也可以使用“javac”命令，如果显示如图 1-5 所示，则表示配置成功。



图 1-5　JDK 环境变量配置验证

1.4　Java 语言开发工具

俗话说“没有金刚钻，别揽瓷器活”，有一个好的“金刚钻”，才能更好更快地做好 Java 这个“瓷器活”。目前已经有很多方便的“金刚钻”可以使用，如 Jcreator、JavaWorkshop、NetBeans、Eclipse，等等，这些开发工具都各有所长，Eclipse 较为常用，下面就对 Eclipse 做简单的介绍。

1.4.1　Eclipse 的安装与使用

本书选择 Eclipse 3.2 进行介绍，其中包括 Eclipse 的安装、语言包的安装和 Lomboz 插件的安装。

1．安装 Eclipse

Eclipse 可以从官方网站（www.Eclipse.org）下载到，其安装也非常简单，只要将下载到的 ZIP 压缩文件包解压缩，运行文件夹中的 Eclipse.exe 文件即可。Eclipse 的安装不需要在系统中写入任何文件和注册信息。

说明：Eclipse 的安装路径可以任意选择，如 F:\Eclipse.2\Eclipse.exe。

在 Eclipse 第一次运行时，会弹出一个对话框，提示选择工作空间，如图 1-6 所示，选中图中标示的复选框，可以将当前路径设置为默认工作空间，然后单击“OK”按钮，开始运行 Eclipse，出现欢迎界面（如图 1-7 所示）。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

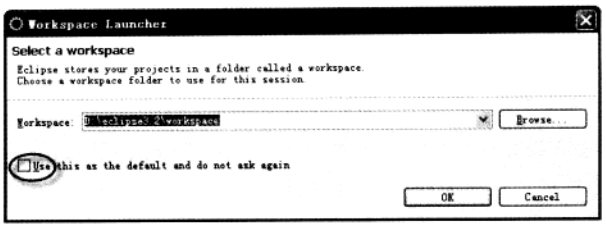


图 1-6　选择工作空间的对话框

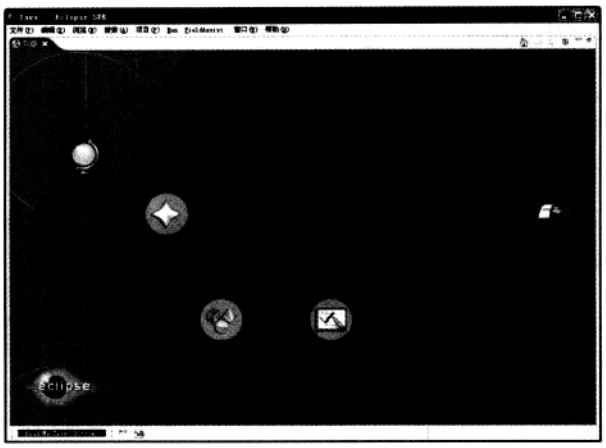


图 1-7　Eclipse 的欢迎界面

说明：Eclipse 的所有项目源代码将存放在工作空间，同时存放一些 Eclipse 的配置文件。若不选中该复选框，则每次启动时都会弹出该对话框。

创建了工作空间后，每一个项目都会在工作空间内创建一个以项目名命名的文件夹。工作空间文件夹中有一个名为“.metadata”的文件夹，用于存放项目信息和 Eclipse 的配置文件，如果此文件夹被删除，就需要手动导入项目，工作空间下就会重新生成一个“.metadata”文件夹。

注意：启动 Eclipse 时弹出欢迎界面，可以查看工作空间是否被修改或删除了。如果在开启 Eclipse 后出现项目的丢失，建议查看工作空间内项目是否被移除，或者配置文件夹是否被删除。

2. 安装 Eclipse 语言包

从官方网站下载的 Eclipse 是纯英文版的，如果提供给不懂英语的人使用，就需要安装一个语言包。

语言包就是 Eclipse 的翻译，也叫做多国语言包，有多种语言的翻译，根据 Windows 系统的语言环境自动选择。

语言包可以在官方网站（www.Eclipse.org）下载，创建一个目录“D:\Eclipse 3.2\eclipse_plugins”，用于专门存放 Eclipse 的插件，并在此目录下创建一个名为“language”的文件夹，用来存放语言包。将下载到的语言包解压缩到 language 文件夹中，新建一个名为“links”

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

文件夹，在“links”文件夹下创建文件“language_zn.link”并用记事本打开，打开后输入如下内容并保存。

```
path= D:/Eclipse3.2/eclipse_plugins/language
```

说明：这里路径分隔符可以使用双斜线，即 D:\\Eclipse3.2\\eclipse_plugins\\language。可以使用绝对路径，也可以使用相对路径。

在进行完以上操作后，重新启动 Eclipse，就会成功汉化（如图 1-8 所示），因为 Eclipse 启动时，会查找 link 文件，并根据配置信息来调用语言包，因此如图 1-8 所示，在重新开启 Eclipse 后，菜单选项都变成了中文。

注意：如果没有成功汉化，请查看文件路径是否正确，或者语言包的版本是否和 Eclipse 的版本相同，例如，Eclipse 3.2 的语言包就无法将 Eclipse 3.0 汉化。Eclipse 的汉化也和操作系统的版本有关，请查看操作系统是否是中文版。

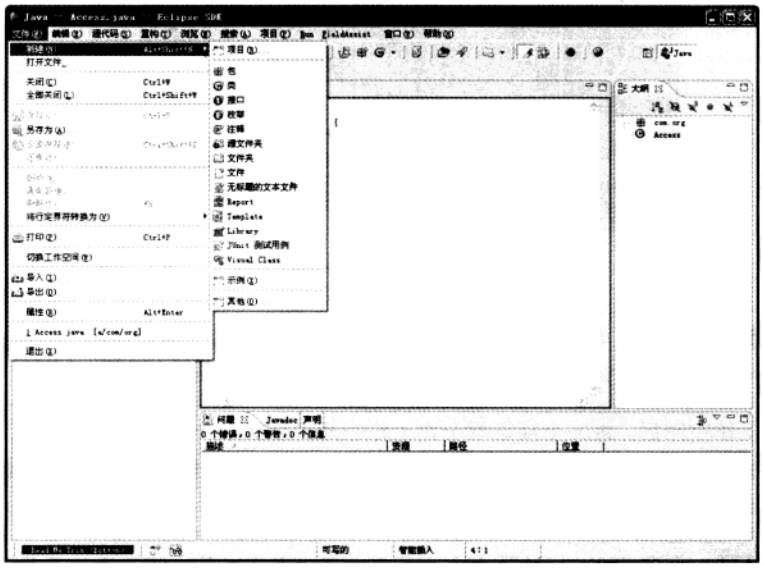


图 1-8　中文界面

说明：如果想要将中文版换成英文版，只需要将 link 文件删除，如果在启动的时候找不到 link 文件，Eclipse 就不会加载语言包，从而还原回英文版，开启 Eclipse 时会出现错误提示信息，只需要再重新启动一次即可。

在 link 文件中可以配置多个注册信息，设定多个 path（每个分成一行）。建议每个配置单独创建一个 link 文件，便于插件的管理。

1.4.2　MyEclipse 的安装

MyEclipse 是近几年发展起来的比较成熟的 Java EE 开发工具，以 MyEclipse 插件的形式运行在 Eclipse 平台上。MyEclipse 在 Java EE 开发中拥有绝对优势的市场占有率。MyEclipse 的

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

官方网站是 www.MyEclipseIDE.com，目前最新版本为 MyEclipse 8.6。

MyEclipse 集成了 Spring、Struts、Hibernate 等各种框架，并且可以自动将 Web 项目部署到 Tomcat、JBoss 等服务器，可以浏览 MySQL、Oracle 等数据库，功能十分强大，是一款非常理想的 JavaWeb 开发工具。

安装 MyEclipse 需要选择 MyEclipse 所要安装的路径（如图 1-9 所示），其余的可以直接单击“下一步”按钮。

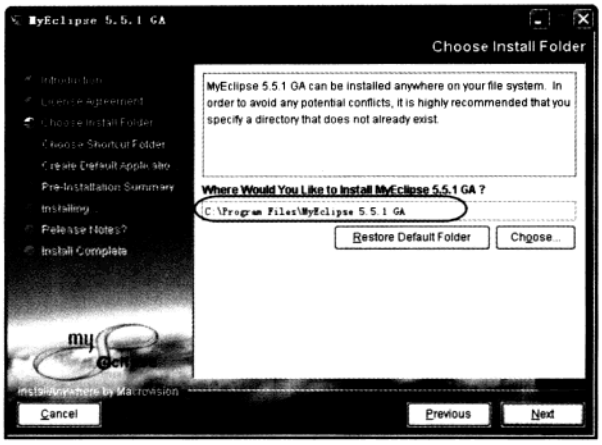


图 1-9　选择安装路径

MyEclipse 安装完成后，从“开始”菜单启动，在首次启动 MyEclipse 时，会让用户选择工作空间（如图 1-10 所示）。

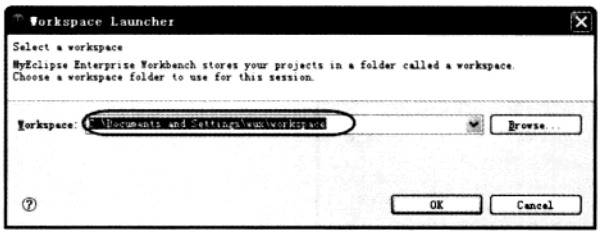


图 1-10　配置工作空间

1.4.3　使用开发工具编写“HelloWorld!”

前面已经了解了开发工具，在本小节中就使用开发工具来完成一个简单的程序，在控制台中输出“HelloWorld!”，主要步骤如下。

① 右击“包资源管理器”空白处，选择“New”→“Project”菜单项，弹出“New Project”对话框（如图 1-11 所示），选择“Java Project”选项，单击“Next”按钮。

② 弹出“New Java Project”对话框（如图 1-12 所示），在“Project name”文本框中输入新建项目的名称，如“ch01”，然后单击“Finish”按钮，可以完成项目的创建。

③ 创建一个类，在类中写入输出语句。右击项目名“ch01”，选择“New”→“Class”菜单项，弹出“New Java Class”对话框，在“Name”文本框中输入类名“HelloWorld”，单击“Finish”按钮，如图 1-13 所示，完成创建。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

④ 类创建完成后，在其中写入代码，以完成打印输出的功能。在 Java 中，打印输出语句是“System.out.println()”，HelloWorld 类的完整代码如下。

【范例 1-1】示例代码 1-1 是一个打印输出“HelloWorld!”的程序。

示例代码 1-1

```
01  public class HelloWorld {
02      public static void main(String[] args) {
03          System.out.println("HelloWorld!");
04      }
05  }
```

【运行结果】运行结果如图 1-14。



图 1-11　新建项目

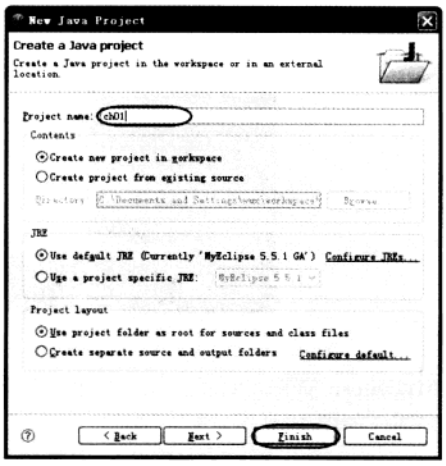


图 1-12　填写项目名

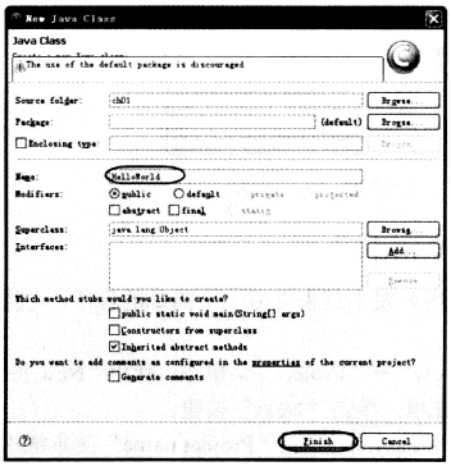


图 1-13　创建类“HelloWorld”

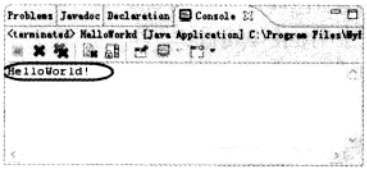


图 1-14　运行结果

【程序解析】在程序中创建了一个 main 方法，然后在该方法中输入打印输出语句，并打印输出“HelloWorld!”文本。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



1.5　习题

【习题 1-1】Java 语言的特点主要有哪些？
【习题 1-2】如何配置 Java 环境变量？
【习题 1-3】如何测试 JDK 是否安装成功？
【习题 1-4】了解常用的 Java 开发工具有哪些？
【习题 1-5】试着写一个简单的 Java 程序？

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第 2 章　Java 基本数据类型

本章主要介绍使用 Java 语言编程需要掌握的基础知识，包括基本数据类型及它们之间的转换、运算符和表达式等内容。



2.1　基本数据类型

在 Java 中，基本数据类型就相当于汉语中的拼音，这些数据类型是构成其他数据类型的基础。基本数据类型主要包括布尔类型、字符型、整型、实型。

2.1.1　布尔类型

人小的时候，还没有学会说话，那所有的事情都需要用动作来说明，例如，不喜欢橘子，则在妈妈喂橘子吃的时候使劲摇头，表示“否”，而若喜欢，则会吃掉或者点头，表示“是”。在 Java 中进行逻辑判断的是布尔类型。

布尔类型用关键字 Boolean 来表示，布尔类型的值只有 true 和 false 两个。布尔数据类型较多使用在程序的流程控制语句中。该类型的默认值是 false，也就是说如果定义了一个布尔类型的变量，但是并没有给该变量赋值，则默认的该变量值是 false。以下示例测试了布尔类型数据。

【范例 2-1】示例代码 2-1 是一个测试布尔类型数据的程序。

示例代码 2-1

```
01  public class BooleanTest {
02      static boolean zhangsan;
03      static boolean lisi=true;
04      public static void main(String[] args) {
05
06          System.out.println("张三: "+zhangsan);
07          System.out.println("李四: "+lisi);
08
09          if(zhangsan){
10              System.out.println("张三有时间..");
11          }
12          if(lisi){
13              System.out.println("李四有时间..");
14          }
15      }
16  }
```

【运行结果】程序运行结果如下。

```
张三: false
李四: true
李四有时间..
```

【代码解析】在程序中，首先创建了两个布尔类型的成员变量 zhangsan 和 lisi，并为 lisi 赋值为 true。创建一个主方法，并在主方法中分别打印输出两个成员变量的值，接下来使用 if 语句进行判断，如果 zhangsan 值为 true，则会打印“张三有时间..”，如果 lisi 的值为 true，则会打印输出“李四有时间..”（第 4～13 行）。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



2.1.2　字符型

汉语拼音由 26 个基本的字母组成，每一个字母都是一个字符型数据。字符型数据类型表示单一的字符，并且使用关键字 char 来表示。Java 中的字符型数据属于 Unicode 编码。下面是一个字符型数据的测试程序。

【范例 2-2】示例代码 2-2 是一个测试字符型数据的程序。

示例代码 2-2

```
01  public class CharTest {
02      public static void main(String[] args) {
03          char c;
04          c='c';
05          System.out.println("字符型数据的值是："+c);
06      }
07  }
```

【运行结果】程序运行结果如下。

```
字符型数据的值是：c
```

【代码解析】在程序中，创建一个主方法（第 2 行），并在主方法中创建一个字符类型变量 c 并赋值为“c”，然后打印输出“c”（第 3～5 行）。

2.1.3　整型

整型数据就是整数，表示不含有小数部分的数字。在 Java 语言中，一共有四种整数类型，分别是 byte、short、int 和 long。这四种整数类型所占用的空间和取值范围不同。

- byte：取值范围是-2^7～2^7-1，默认值是 0，所占用的空间是 1 个字节。
- short：取值范围是-2^{15}～2^{15}-1，默认值是 0，所占用的空间是 2 个字节。
- int：取值范围是-2^{31}～2^{31}-1，默认值是 0，所占用的空间是 4 个字节。
- long：取值范围是-2^{63}～2^{63}-1，默认值是 0，所占用的空间是 8 个字节。

在 Java 中，一个普通的整型数数字默认为 int 类型，如果需要使用 long 类型，需要在数字尾部加“L”，如 20L。

【范例 2-3】示例代码 2-3 是一个测试整型数据的程序。

示例代码 2-3

```
01  public class IntTest {
02      public static void main(String[] args) {
03          //定义各类型变量
04          byte m;
05          short x;
06          int i;
07          long l;
08          //为各类型变量赋值
09          m=12;
10          x=1;
11          i=2;
12          l=1000;
13          //打印输出各类型变量的值
14          System.out.println("byte 类型变量 m 的值为："+m);
15          System.out.println("short 类型变量 x 的值为："+x);
16          System.out.println("int  类型变量 i 的值为："+i);
17          System.out.println("long 类型变量 l 的值为："+l);
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
18        }
19    }
```

【运行结果】程序的运行结果如下。

```
byte 类型变量 m 的值为：12
short 类型变量 x 的值为：1
int  类型变量 i 的值为：2
long 类型变量 l 的值为：1000
```

【代码解析】在程序中，首先创建一个主方法（第 2 行），在主方法中分别创建四种整型数据类型变量 m、x、i、l，并分别为各个变量赋值（第 4～12 行），然后分别将这几个变量进行打印输出（第 14～17 行）。

2.1.4 实型

含有小数的数字类型就是实型，实型又叫做浮点型。在 Java 中共有 float 和 double 两种实型数据类型。

float 称为单精度浮点型，double 称为双精度浮点型。二者之间的区别在于双精度浮点型比单精度浮点型的取值范围大，而且有更高的精度。当然，拥有大的取值范围和精度的同时，也占用了大的内存空间。

【范例 2-4】示例代码 2-4 是一个测试实型数据的程序。

示例代码 2-4

```
01  public class FloatTest {
02      public static void main(String[] args) {
03          //声明两个变量
04          float f;
05          double d;
06          //分别为两个变量赋值
07          f=12.11f;
08          d=1.001;
09          //分别将两个值打印输出
10          System.out.println("float 型的变量 f 的值是："+f);
11          System.out.println("double 型变量 d 的值是："+d);
12      }
13  }
```

【运行结果】程序的运行结果如下。

```
float 型的变量 f 的值是：12.11
double 型变量 d 的值是：1.001
```

【代码解析】在程序中，首先分别定义了一个单精度浮点型和双精度浮点型变量（第 4～5 行），然后分别为这两个变量赋值（第 7～8 行），最后将两个变量的值分别打印输出（第 10～11 行）。

实型的值不会在计算过程中因为溢出而导致异常。为了避免抛出异常，在超出数值范围溢出时，Java 定义了特殊的浮点数来进行表示，这种处理方法，使得 Java 大大提高了本身的健壮性，而且也简化了程序。

2.2 数据类型的转换

在程序的编写过程中，不同数据类型之间的变量经常需要互相赋值或互相运算，这就需要

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



进行数据类型的转换。各种数据类型要根据分配空间的大小、取值范围的不同来划分优先级，然后再根据优先级来进行数据的转换。

2.2.1　隐式数据类型转换

去银行进行货币兑换时，只需要交上需要兑换的货币，银行就会自动根据基数来进行货币的兑换。隐式数据类型转换也有些类似，它是自动由低优先级到高优先级数据的转换赋值，不需要进行特别的说明。

【范例 2-5】示例代码 2-5 是一个测试数据类型转换的程序。

示例代码 2-5

```
01  public class TypeTest {
02      public static void main(String[] args) {
03          //声明各种类型变量并赋初值
04          byte b=1;
05          short s=2;
06          char c='c';
07          int i=3;
08          long l=4;
09          float f=5;
10          double d=6.0;
11          System.out.println("变量 b 的值为："+b);
12          System.out.println("变量 s 的值为："+s);
13          System.out.println("变量 c 的值为："+c);
14          System.out.println("变量 i 的值为："+i);
15          System.out.println("变量 l 的值为："+l);
16          System.out.println("变量 f 的值为："+f);
17          System.out.println("变量 d 的值为："+d);
18          //分别将各个类型进行转换
19          s=b;
20          i=c;
21          i=s;
22          l=i;
23          f=l;
24          d=f;
25          d=b;
26          d=f+d;
27          System.out.println("转换后变量 b 的值为："+b);
28          System.out.println("转换后变量 s 的值为："+s);
29          System.out.println("转换后变量 c 的值为："+c);
30          System.out.println("转换后变量 i 的值为："+i);
31          System.out.println("转换后变量 l 的值为："+l);
32          System.out.println("转换后变量 f 的值为："+f);
33          System.out.println("转换后变量 d 的值为："+d);
34
35      }
}
```

【运行结果】程序运行结果如下。

```
变量 b 的值为：1
变量 s 的值为：2
变量 c 的值为：c
变量 i 的值为：3
变量 l 的值为：4
变量 f 的值为：5.0
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
变量 d 的值为：6.0
转换后变量 b 的值为：1
转换后变量 s 的值为：1
转换后变量 c 的值为：c
转换后变量 i 的值为：1
转换后变量 l 的值为：1
转换后变量 f 的值为：1.0
转换后变量 d 的值为：2.0
```

【代码解析】在程序中，首先创建了几个不同类型的变量（第 4～10 行），接着为各个变量进行赋值并打印输出（第 11～17 行），然后将各个变量进行类型的转换并运行一个特殊的加法运算，在执行该运算之前，f 需要自动转换为 double 类型（第 19～26 行），最后将各个转换后的变量打印输出（第 27～33 行），并与转换以前的变量进行对比，查看值的变化。

由以上的程序可以看出，低优先级类型向高优先级类型的赋值是自动完成的。在进行数值运算时，不同数据类型的操作数首先会转换成相同的数据类型，然后再进行计算。

2.2.2 显式类型转换

显式类型转换也就是指优先级高的类型向优先级低的类型进行转换时，需要进行强制类型转换。

【范例 2-6】示例代码 2-6 是一个测试数据类型转换的程序。

示例代码 2-6

```
01  public class Qiangzhi {
02      public static void main(String[] args) {
03          //声明两个变量
04          int i=111;
05          //强制转换
06          byte b=(byte)i;
07          System.out.println("int 类型强制转换成 byte 后的值是："+b);
08          double d=111.222;
09          int j=(int)d;
10          System.out.println("double 强制转换为 int 后的值是："+j);
11      }
12  }
```

【运行结果】程序运行结果如下。

```
int 类型强制转换成 byte 后的值是：111
double 强制转换为 int 后的值是：111
```

【代码解析】在程序中，首先声明两个变量，然后将 int 类型强制转换为 byte 类型，由于是整型数据，所以采用截取的方式来进行转换，这是由二进制的表示方法决定的。然后，将 double 类型强制转换为 int 类型。最后输出二者的转换结果。

2.3 运算符和表达式

刚进小学进行学习的时候，首先要学习的是加、减、乘、除等法则，还有一些相对应的符号，这些符号就是运算符，如果要进行运算，还有列出等式，这就是表达式。在程序中，运算符和操作数组成表达式来完成相应的运算。而不同的运算符和操作数又会组成不同的表达式。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



2.3.1　算术运算符及表达式

算术运算符就是在小学时所学到的加减乘除运算。算术运算符就是进行加、减、乘、除运算的符号，对应的运算符是“+”、“-”、“*”、“/”。算术运算符实现了整型数据类型和浮点数据类型之间的计算。

【范例 2-7】示例代码 2-7 是一个测试算术运算符的程序。

示例代码 2-7

```
01  public class NumericOperatorTest {
02      public static void main(String[] args) {
03          //声明变量并进行运算
04          int i=1+2;
05          int j=5-1;
06          int k=5*6;
07          int l=6/5;
08          //打印输出运算结果
09          System.out.println("相加结果是: "+i);
10          System.out.println("相减结果是: "+j);
11          System.out.println("相乘结果是: "+k);
12          System.out.println("相除结果是: "+l);
13
14      }
15  }
```

【运行结果】程序的运行结果如下。

```
相加结果是: 3
相减结果是: 4
相乘结果是: 30
相除结果是: 1
```

【代码解析】在以上的程序中，首先创建了四个变量，并分别进行“+”、“-”、“*”、“/”的运算，然后分别将各个运算的值进行打印输出。

在 Java 中，还有递增和递减操作，分别使用“++”、“—”来表示，“++”表示值加 1，“—”表示值减 1。

【范例 2-8】示例代码 2-8 是一个测试算术运算符的程序。

示例代码 2-8

```
01  public class IDOperatorTest {
02      public static void main(String[] args) {
03          //定义四个变量，并使用递加
04          int a=1;
05          int b=1;
06          int c=a++;
07          int d=b++;
08          //将结果分别打印输出
09          System.out.println("a 的值是: "+a);
10          System.out.println("b 的值是: "+b);
11          System.out.println("c 的值是: "+c);
12          System.out.println("d 的值是: "+d+"\n");
13          //定义四个变量，并使用递减
14          int e=5;
15          int f=5;
16          int g=e--;
17          int h=f--;
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
18            //将结果分别打印输出
19            System.out.println("e 的值是: "+e);
20            System.out.println("f 的值是: "+f);
21            System.out.println("g 的值是: "+g);
22            System.out.println("h 的值是: "+h);
23
24        }
25    }
```

【运行结果】程序的运行结果如下。

```
a 的值是: 2
b 的值是: 2
c 的值是: 1
d 的值是: 1

e 的值是: 4
f 的值是: 4
g 的值是: 5
h 的值是: 5
```

【代码解析】在程序中，首先创建了四个变量并赋予了初值，后面的两个变量使用了递加的方法来计算赋值（第 4～7 行）。然后，分别将结果打印输出（第 9～12 行）。接着，再创建四个变量并赋初值，后面的两个变量使用了递减的方法来计算赋值（第 14～22 行）。

2.3.2 关系运算符及表达式

程序中，操作数之间有时会使用到关系运算符，用于判断大小或真假。

关系运算符适用于判断大小或者判断真假，返回值为布尔类型，如判断相等的运算符“==”、判断不等的运算符“!=”。关系运算符通常和逻辑条件语句配合使用。关系运算的结果只有两种，即真（true）和假（false）。Java 中的关系运算符有六种，如下所示。

- >：表示大于，如 X>Y，若 X 的值大于 Y 的值，则为真，否则为假。
- <：表示小于，如 X<Y，若 X 的值小 Y 的值，则为真，否则为假。
- >=：大于等于，如 X>=Y，若 X 的值大于或等于 Y 的值，则为真，否则为假。
- <=：小于等于，如 X<=Y，若 X 的值小于或等于 Y 的值，则为真，否则为假。
- ==：等于，如 X==Y，若 X 与 Y 的值相等，则为真，否则为假。
- !=：不等于，如 X!=Y，若 X 与 Y 的值不相等，则为真，否则为假。

【范例 2-9】示例代码 2-9 是一个测试关系运算符的程序。

示例代码 2-9

```
01  public class Validate {
02      public static void main(String[] args) {
03          System.out.println("5>4 的值是: "+(5>4));
04          System.out.println("5<4 的值是: "+(5<4));
05          System.out.println("5>=4 的值是: "+(5>=4));
06          System.out.println("5<=4 的值是: "+(5<=4));
07          System.out.println("5==4 的值是: "+(5==4));
08          System.out.println("5!=4 的值是: "+(5!=4));
09
10      }
11  }
```

【运行结果】程序的运行结果如下。

```
5>4 的值是: true
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
5<4 的值是：false
5>=4 的值是：true
5<=4 的值是：false
5==4 的值是：false
5!=4 的值是：true
```

【代码解析】在程序中，依次输出各个关系运算表达式所对应的值（第 3～8 行），从以上的程序可以发现，两个数字相等或者前者大于后者时，使用>=运算符都会返回 true，而两个数字相等或者前者小于后者时，使用<=运算符都会返回 false。

2.3.3　逻辑运算符及表达式

逻辑运算符的计算对象是布尔变量，当需要同时对多个条件进行判断时，需要使用逻辑与符号“&&”、逻辑或符号“||”等，Java 中共有六种逻辑运算符。

- &：逻辑与，如 x&y，表示 x 和 y 都真则真。
- |：逻辑或，如 x|y，表示 x 和 y 中有一个为真则真。
- ！：逻辑非，如！x，表示 x 为真则假，x 为假则真。
- &&：逻辑与，如 x&&y，表示 x 和 y 都真则为真。
- ||：逻辑或，如 x||y，表示 x 和 y 中有一个为真则为真。
- ^：异或，如 x^y，表示 x 和 y 都为真或者都为假时，为假，否则为真。



注意：“&&”、“||”与“&”、“|”的区别是：“&&”和“||”是短路逻辑操作符，即如果根据操作符左边的 Boolean 值就能推知整个表达式的值，就不会再计算操作符右边的 Boolean 值了；而非短路操作符，即“&”和“|”，就是无论如何都要把两边的值都计算出来。

【范例 2-10】示例代码 2-10 是一个测试逻辑运算符的程序。

示例代码 2-10

```
01  public class LogicalOperator {
02      public static void main(String[] args) {
03          //声明变量 x 和 y，并赋初始值
04          boolean x=true;
05          boolean y=false;
06          //分别打印输出逻辑运算结果
07          System.out.println("x&y 的值是："+(x&y));
08          System.out.println("x|y 的值是："+(x|y));
09          System.out.println("!x 的值是："+!x);
10          System.out.println("x&&y 的值是："+(x&&y));
11          System.out.println("x||y 的值是："+(x||y));
12          System.out.println("x^y 的值是："+(x^y));
13          System.out.println("x^x 的值是："+(x^x));
14
15      }
16  }
```

【运行结果】程序的运行结果如下。

```
x&y 的值是：false
x|y 的值是：true
!x 的值是：false
x&&y 的值是：false
x||y 的值是：true
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
x^y 的值是：true
x^x 的值是：false
```

【代码解析】在程序中，首先声明了两个布尔类型变量 x 和 y（第 4 和 5 行），然后运用逻辑运算符对这两个值进行逻辑判断，并依次打印输出逻辑判断的值（第 7～13 行）。

“&&”和“&”、“||”和“|”的判断值结果相同，但是它们之间还是有区别的，“&&”和“||”只要计算完左边的值就可以确定整个表达式的值，而不必再计算右边的值，而“&”和“|”需要将两边的结果全部计算完毕后，才可以计算结果值。

2.3.4 位运算符及表达式

位运算符是指按二进制位的形式对操作数进行运算，在位运算中，操作数必须是整数类型。位运算符共有七个，分别是>>（位右移）、<<（位左移）、>>>（无符号位右移）、&（按位与）、|（按位或）、^（按位异或）、～（按位取反）。

- >>：位右移，用于将第一个操作数的比特位向右移动第二个操作数指定的位数。
- <<：位左移，用于将第一个操作数的比特位向左移动第二个操作数指定的位数。
- >>>：无符号位右移运算符，无符号右移的规则和右移的规则是相同的，只是在填充时，不管原数是整数还是负数，都是用 0 来填充。
- &：按位与，同&&的运算类似，只不过操作数变成 0 和 1。
- |：按位或，同||的运算类似。
- ^：按位异或，如果两个操作数的值相同，结果为 0，否则结果为 1。
- ～：按位取反，为一元运算符，将二进制取反，即 1 变 0、0 变 1。

【范例 2-11】示例代码 2-11 是一个测试位运算符的程序。

示例代码 2-11

```
01  public class PositionTest {
02
03  public static void main(String[] args) {
04      //使用位运算>>
05      System.out.println("00001111>>2 运算结果是："+(00001111>>2));
06      //使用位运算>>
07      System.out.println("11000011>>2 运算结果是："+(11000011>>2));
08      //使用位运算<<
09      System.out.println("00001111<<2 运算结果是："+(00001111<<2));
10      //使用位运算<<
11      System.out.println("11000011<<2 运算结果是："+(11000011<<2));
12      //使用位运算&
13      System.out.println("00001111&00001111运算结果是："+(00001111&0000111));
14      //使用位运算|
15      System.out.println("00001111|00001111运算结果是:"+(00001111|00001111));
16      //使用位运算^
17      System.out.println("00001111^00001111运算结果是:"+(00001111^00001111));
18      //使用位运算~
19      System.out.println("~ 00001111 运算结果是："+(~ 00001111));

    }
}
```

【运行结果】程序运行结果如下。

```
00001111>>2 运算结果是：146
11000011>>2 运算结果是：2750002
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
00001111<<2 运算结果是：2340
11000011<<2 运算结果是：44000044
00001111 & 00001111 运算结果是：73
00001111 | 00001111 运算结果是：585
00001111 ^ 00001111 运算结果是：0
~ 00001111 运算结果是：-586
```

【代码解析】在以上程序中，以二进制的形式给出了位运算的举例，在实际程序运行中，底层计算机将位运算转换成二进制代码进行运算，所以位运算比较接近计算机的底层，并且效率较高。

2.3.5　赋值运算符及表达式

赋值就像是给一个刚出生的小孩起名一样，将这个孩子和这个名字中间画上等号。赋值运算符只有一个，就是“=”。赋值运算符用于将运算符右边的值赋给运算符左边的变量。

【范例 2-12】示例代码 2-12 是一个测试赋值运算符的程序。

示例代码 2-12

```
01  public class OperatorTest {
02      public static void main(String[] args) {
03          int  x,y;
04          x=1;                    //将 1 赋值给 x
05          y=x*10;                 //将表达式的值赋值给 y
06
07          System.out.println("x 的值是："+x);
08          System.out.println("y 的值是："+y);
09
10      }
11  }
```

【运行结果】程序运行结果如下。

```
x 的值是：1
y 的值是：10
```

【代码解析】在以上的程序中，声明了两个 int 类型的变量 x 和 y，首先将 x 赋值为 1，然后将 y 的值设为 x 的 10 倍，最后输出 x 和 y 的值。

2.4　习题

【习题 2-1】对几种类型进行转换，发生的条件是：两种类型是兼容的，或者目标类型的范围比原类型的范围大。

【分析】该题目主要考察的是基本数据类型的隐式转换，首先要创建各种基本的数据类型，然后由低优先级到高优先级进行数据转换。

【核心代码】

```
byte b = 12;          //声明一个字节型变量
char c = 'b';         //声明一个字符型变量
short s = 13;         //声明一个短整型变量
int i = 567;          //声明一个整型变量
long l = 678756;      //声明一个长整型变量
float ft = 23.67f;    //声明一个单精度浮点型变量
double db = 6.124d;   //声明一个双精度浮点型变量
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
double result = (ft * b) + (l * ft) + (i / c) - (db * s); //result 声明为其他类型
                                                          //会出错,除非进行类型转换
System.out.print(" 结果 : " + result+" = ");              //将结果输出
System.out.println((ft * b) + " " + " + (l * ft) + " + " + (i / c) + " - "
            + (db * s));                                  //输出经过运算获得的结果
```

【习题 2-2】在不同的数据类型之间进行转换。

【分析】该题目是用来考查读者对基本数据类型转换问题的理解，练习在创建的几种不同的基本数据类型之间进行转换。

【核心代码】

```
char c = 'a';              //声明一个字符型变量
byte b = 44;               //声明一个字节型变量
short s0 = b;              //声明一个短整型变量
//声明一个整型变量
int i0 = s0;
int i1 = c;
long l = i0;               //声明一个长整型变量
float f = l;               //声明一个单精度浮点型变量
double d = f;              //声明一个双精度浮点型变量
float fl = 1.7f;           //声明一个单精度浮点型变量
double db = fl;            //声明一个双精度浮点型变量
// 一个数从一种类型转换成另外一种类型，再转换回来时，值还是一样。
System.out.println("fl = " + fl + "; dou = " + db);
fl = (float) db;           //将 double 型强制转换成 float 型
System.out.println("fl = " + fl + "; dou = " + db);
```

【习题 2-3】编写一个程序，输出变量的初始值。

【分析】该题目主要考查的是用不同的基本数据类型数据为变量赋值，首先定义几个不同基本数据类型的变量，然后为其赋值并分别将其输出，以查看该变量的初始值。

【核心代码】

```
byte m;
short x;
int i;
long l;
//为各类型变量赋值
m=12;
x=1;
i=2;
l=1000;
//打印输出各类型变量的值
System.out.println("byte 类型变量 m 的值为："+m);
System.out.println("short 类型变量 x 的值为："+x);
System.out.println("int  类型变量 i 的值为："+i);
System.out.println("long 类型变量 l 的值为："+l);
```

【习题 2-4】设 n 为自然数，分别求出 n=0，2，4，6，8 的阶乘。

【分析】该题主要考察的是算术运算符及表达式的使用，Factorial 方法递归调用自身，进行阶乘运算。

【核心代码】

```
public static void main(String[] args) {
        Factorial ff=new Factorial();
        for(int i=0;i<5;i++){
            ff.setInitVal(2*(i+1));
            ff.result= Factorial(ff.initVal);
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



```
            ff.print();
        }

    }
    public static int Factorial(int n){

        if(n==0)return 1;
        return n* Factorial(n-1);
    }
```

【习题 2-5】判断关系表达式的真假。

【分析】该题主要考查的是关系运算符和表达式，输出几个关系表达式的真假判断结果。

【核心代码】

```
System.out.println("5>6 的值是：" + (5>6));
System.out.println("5<6 的值是：" + (5<6));
System.out.println("5>=6 的值是：" + (5>=6));
System.out.println("5<=6 的值是：" + (5<=6));
System.out.println("5==6 的值是：" + (5==6));
System.out.println("5!=6 的值是：" + (5!=6));
```

【习题 2-6】判断逻辑表达式的值。

【分析】该题主要考查的是逻辑运算符和表达式，首先定义变量，然后输出逻辑表达式的值。

【核心代码】

```
public class LogicalOperator {
   public static void main(String[] args) {
      boolean x=true;
      boolean y=false;
      System.out.println("x&y 的值：" + (x&y));
      System.out.println("x|y 的值：" + (x|y));
      System.out.println("x^y 的值：" + (x^y));
      System.out.println("x^x 的值：" + (x^x));
      System.out.println("x&&y 的值：" + (x&&y));
      System.out.println("x||y 的值：" + (x||y));
      System.out.println("!y 的值：" + (!y));
   }
}
```

【习题 2-7】计算位运算表达式的值。

【分析】该题主要考查的是位移运算符和表达式，输出位运算表达式的值。

【核心代码】

```
public class ShiftOperator {
   public static void main(String[] args) {
      System.out.println(Integer.toBinaryString(789));
      System.out.println(Integer.toBinaryString(-789));
      System.out.println(789>>3);
      System.out.println(Integer.toBinaryString(789>>3));
      System.out.println(-789>>3);
      System.out.println(Integer.toBinaryString(-789>>3));
      System.out.println(789<<3);
      System.out.println(Integer.toBinaryString(789<<3));
      System.out.println(-789<<3);
      System.out.println(Integer.toBinaryString(-789<<3));
      System.out.println(789>>>3);
      System.out.println(Integer.toBinaryString(789>>>3));
      System.out.println(-789>>>3);
      System.out.println(Integer.toBinaryString(-789>>>3));
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
    }
}
```

【习题 2-8】声明几个数字变量，进行算术运算，并输出计算结果。

【分析】该题主要考查的是算术运算符和表达式，首先定义几个变量并赋值，然后使用算术运算表达式为变量赋值，并输出结果。

【核心代码】

```
public class DoSums {
   public static void main(String[] args) {
      int a = 31;
      float b = 32;
      long c = 33;
      double d = 34;

      double s = (a * b + c*d)/2;
      System.out.println("(a * b + c*d)/2 的值是："+ s);
   }
}
```

【习题 2-9】写一个程序，求两个数字的乘积，如果大于 0，说明两个数字同号，如果小于 0，说明两个数字异号，如果等于 0，说明其中一个数字是 0。

【分析】该题主要考查的是关系运算符和表达式。

【核心代码】

```
public class Multiply {
   public static void main(String[] args) {
      Scanner reader = new Scanner(System.in);
      System.out.println("请输入数字：");

      int x =  reader.nextInt();
      int y =  reader.nextInt();
      int z = x * y;
      if (z > 0) {
         System.out.println("两个数字同号");
      } else if (z < 0) {
         System.out.println("两个数字异号");
      } else {
         System.out.println("两个数字中有一个是 0");
      }
   }
}
```

【习题 2-10】计算逻辑运算表达式的值。

【分析】该题主要考查的是逻辑运算符和表达式。

【核心代码】

```
public class DoBoolean {
   public static void main(String[] args) {
      Scanner reader = new Scanner(System.in);
      System.out.println("请输入逻辑：");
      boolean a = reader.nextBoolean();
      boolean b = reader.nextBoolean();
      System.out.println("a && b 的值是：" + (a && b));
      System.out.println("a || b 的值是：" + (a || b));
      System.out.println("(a && b) && (a || b)的值是：" + ((a && b) && (a || b)));
      System.out.println("(a && b) || (a || b)的值是：" + ((a && b) || (a || b)));
   }
}
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第 3 章　数　　组

数组是 Java 中经常使用到的一项工具，它是用来存放具有相同数据类型的数值的数据结构。例如，将几个标上数字的小球放到一个罐子中，那这个罐子就可以看做是一个数组，其中的小球就是数组中的元素。



3.1　数组基础

Java 中的数组可以看做是一种特殊的对象，也就是把数组看做是一个同种类型变量的集合，在同一个数组中的所有数据都有相同的数据类型，并且使用统一的数组名，通过下标来区分数组中各个不同的元素。

3.1.1　创建数组

数组在使用以前需要进行声明。数组的声明共有两种形式，格式如下。

```
ArrayType ArrayName[];
ArrayType[] ArrayName;
```

上面格式中的“[]”符号表明这是一个数组。其中，第二种声明方式可以同时对多个数组进行声明。声明一个整型数组的示例如下。

```
int array[];
int[] array1,array2;
```

上面的语句只是对数组进行了声明，还没有为该数组分配内存空间，所以，此时不能对数组进行访问。可以使用 new 来为数组分配内存空间，格式如下。

```
array1 =new int[5];
```

在分配完内存空间后，数组中就有了以下 5 个元素。

```
array[0]
array[1]
array[2]
array[3]
array[4]
```



注意：在 Java 的数组中，下标是从 0 开始而不是从 1 开始的，这也就意味着最后一个索引号不是数组的长度，而是一个比数组的长度小 1 的数。

【范例 3-1】示例代码 3-1 是一个测试数组下标的程序。

示例代码 3-1

```
01  public class ArrayTest {
02      public static void main(String[] args) {
03          //创建一个容量为 5 的数组
04          int[] array1=new int[5];
05          //打印输出第五个元素
06          System.out.println(array1[5]);
07
```

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
08        }
09    }
```

【运行结果】程序运行结果如下。

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 5
    at ArrayTest.main(ArrayTest.java:9)
```

【代码解析】在以上的程序中，首先定义了一个大小为 5 的整型数组 array1，然后打印出下标为 5 的元素。在前面已经讲过，该数组的下标最大只能是 4，所以打印出下标为 5 的元素显然是不对的，所以程序就会报出以上的异常。

数组在声明之后，数组中的各个元素就可以被访问了，因为在数组创建时，已经自动给出了相应类型的默认值。

【范例 3-2】示例代码 3-2 是一个测试数组默认值的程序。

示例代码 3-2

```
01  public class ArrayDefaultTest {
02      public static void main(String[] args) {
03          //创建一个 byte 类型的数组
04          byte[] byteArray=new byte[1];
05          //创建一个 char 类型的数组
06          char[] charArray=new char[1];
07          //创建一个 int 类型的数组
08          int[] intArray=new int[1];
09          //创建一个 long 类型的数组
10          long[] longArray=new long[1];
11          //创建一个 float 类型的数组
12          float[] floatArray=new float[1];
13          //创建一个 double 类型的数组
14          double[] doubleArray=new double[1];
15
16          System.out.println("byte 类型的默认值为："+byteArray[0]);
17          System.out.println("char 类型的默认值为："+charArray[0]);
18          System.out.println("int 类型的默认值为："+intArray[0]);
19          System.out.println("long 类型的默认值为："+longArray[0]);
20          System.out.println("float 类型的默认值为："+floatArray[0]);
21          System.out.println("double 类型的默认值为："+doubleArray[0]);
22      }
23
24  }
```

【运行结果】程序的运行结果如下。

```
byte 类型的默认值为：0
char 类型的默认值为：
int 类型的默认值为：0
long 类型的默认值为：0
float 类型的默认值为：0.0
double 类型的默认值为：0.0
```

【代码解析】在程序中，创建了几个不同数据类型的数组（第 3～14 行），然后分别打印输出各个数组的第一个值（第 16～21 行），用于测试数组的默认值。

3.1.2 数组的初始化

对数组进行初始化，可以使用 new 的方式或者使用大括号的方式，这两种方式的格式如下。

```
int array1[]=new int[]{1,2,3,4,5,6,7,8,9,10};
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
int array[]={1,2,3,4,5,6,7,8,9,10};
```

这两种初始化的方法是不同的，前者是先创建一个数组，然后将括号里的值赋值给这个数组，而后者是将括号里的值赋值给一个创建好了的数组。还有一种赋值方法比较麻烦，如下所示。

```
int[] array=new int[5];
array[0]=1;
array[1]=2;
array[2]=3;
array[3]=4;
array[4]=5;
```

上面语句的作用和前面语句的作用相同，数组的各个元素会获得相应的值，如果没有对所有的元素进行赋值，它就会自动被初始化为某一个默认值。

【范例 3-3】示例代码 3-3 是一个测试数组默认值的程序。

示例代码 3-3

```
01  public class ArrayNewTest {
02      public static void main(String[] args) {
03
04          int array1[]={1,2,3,4,5};
05          int array2[]=new int[5];
06          array2[0]=1;
07          array2[1]=2;
08          array2[2]=3;
09          array2[3]=4;
10          array2[4]=5;
11
12          for(int i=0;i<5;i++){
13              System.out.println("数组 array1["+i+"]的值是："+array1[i]);
14              System.out.println("数组 array2["+i+"]的值是："+array2[i]);
15          }
16      }
17  }
```

【运行结果】程序的运行结果如下。

```
数组 array1[0]的值是：1
数组 array2[0]的值是：1
数组 array1[1]的值是：2
数组 array2[1]的值是：2
数组 array1[2]的值是：3
数组 array2[2]的值是：3
数组 array1[3]的值是：4
数组 array2[3]的值是：4
数组 array1[4]的值是：5
数组 array2[4]的值是：5
```

【代码解析】在以上程序中，分别使用了两种不同的方法对数组进行赋值（第 4～10 行），并在最后使用 for 循环将各个数组的值打印出来（第 12～14 行）。

Java 中的数组是一种对象，它只有一个公共的实例变量，即 length 变量，这个变量表示数组的长度。如果知道一个数组，但不知道该数组的长度，可以使用 array.length 方法来获得。在使用 for 循环时，也可以使用该方法。

3.1.3　数组的使用

在进行数组的复制时，可以直接将一个数组变量复制给另一个，这时候数组类型变量都会

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

指向同一个数组，即如果有两个数组 array1 和 array2，复制语句如下。

```
array1=array2;
```

此时的两个数组类型变量都会指向同一个数组，也就是原来 array2 所指向的数组。

【范例 3-4】示例代码 3-4 是一个测试数组复制的程序。

示例代码 3-4

```
01  public class ArrayCopyTest {
02      public static void main(String[] args) {
03          //创建数组
04          int[] array1={1,2,3};
05          int[] array2={4,5,6};
06
07          System.out.println("两个数组的初始值为：");
08          for(int i=0;i<array1.length;i++)
09              System.out.println("array1["+i+"]="+array1[i]);
10          for(int i=0;i<array2.length;i++)
11              System.out.println("array2["+i+"]="+array1[i]);
12          //将 array2 赋值给 array1
13          array1=array2;
14
15          System.out.println("在执行数组的复制后，两个数组的值：");
16          //打印输出复制后的值
17          for(int i=0;i<array1.length;i++)
18              System.out.println("array1["+i+"]="+array1[i]);
19          for(int i=0;i<array2.length;i++)
20              System.out.println("array2["+i+"]="+array1[i]);
21
22          System.out.println("将数组 array2 的一个元素改变");
23          array2[2]=8;
24          System.out.println("array2[2]="+array1[2]);
25          System.out.println("array1[2]="+array1[2]);
26      }
27  }
```

【运行结果】程序的运行结果如下。

```
两个数组的初始值为：
array1[0]=1
array1[1]=2
array1[2]=3
array2[0]=4
array2[1]=5
array2[2]=6
在执行数组的复制后，两个数组的值：
array1[0]=4
array1[1]=5
array1[2]=6
array2[0]=4
array2[1]=5
array2[2]=6
将数组 array2 的一个元素改变
array2[2]=8
array1[2]=8
```

【代码解析】在程序中，首先声明了两个数组 array1 和 array2，并对它们进行了初始化（第 4 和 5 行），输出其中的所有元素值（第 7～11 行），然后执行复制语句，再次打印输出两个数组的值（第 15～20 行），可以看出两个数组的元素值相同了，然后再给 array2 数组的下标为 2 的元素重新赋值，并打印输出 array2 和 array1 数组中下标为 2 的元素值（第 22～25 行），从结

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

果可以看出，二者的值是相同的。

3.2　数组的排序

实际应用中经常会涉及到对数组进行排序的操作，这时候就需要使用数组的排序算法，数组的排序算法也是算法和数据结构中的主要内容。数组的排序方法主要包括选择排序和冒泡排序。

3.2.1　选择排序法

选择排序法是比较简单的排序方法，也比较容易理解，基本思路是对一个长度为 i 的数组进行 i 遍的遍历，第一遍遍历出最大的元素并将其与数组的第一个元素交换位置，然后再来进行第二遍的遍历，再最大的元素与第二个元素进行交换。如此遍历 n 遍之后，就会将所有的元素从大到小排列出来了。

【范例 3-5】示例代码 3-5 是一个测试数组选择排序法的程序。

示例代码 3-5

```
01 public class SortTest {
02   public static void main(String[] args) {
03      int[] intarray = {1, 23, 45, 66, 2, 332, 8, 5};
04      int keyvalue;
05      int index;
06      int temp;
07      System.out.println("排序前的数组元素");
08      for (int a : intarray) {
09         System.out.print(a + " ");
10      }
11      System.out.println();
12      for (int i = 0; i < intarray.length; i++) {
13         index = i;
14         keyvalue = intarray[i];
15         for (int j = i; j < intarray.length; j++) {
16            if (intarray[j] < keyvalue) {
17               index=j;
18               keyvalue = intarray[j];
19            }
20         }
21         temp = intarray[i];
22         intarray[i] = intarray[index];
23         intarray[index] = temp;
24      }
25      System.out.println("排序后的数组：");
26      for (int b : intarray) {
27         System.out.print(b + " ");
28      }
29   }
30}
```

【运行结果】程序运行结果如下。

```
排序前的数组元素
1   23   45   66   2   332   8   5
排序后的数组：
1   2   5   8   23   45   66   332
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

【代码解析】首先在程序中声明了一个数组并输出其排序以前的内容，然后对数组进行选择排序，再输出排序后的结果，和排序前做比较。

3.2.2 冒泡排序法

数组的排序方法除了选择排序法外，还有一种冒泡排序法，又叫做交换排序法，冒泡排序的整个过程是把数组元素中比较小的元素看做是重量较轻的，让它上浮，并且依次从底端进行该操作。

【范例 3-6】示例代码 3-6 是一个测试数组冒泡排序法的程序。

示例代码 3-6

```
01  public class MaopaoTest {
02      public static void main(String[] args) {
03
04          int[] intarray={1,2,33,5,34,67,213,8};
05          System.out.println("排序前的数组：");
06          for(int i=0;i<intarray.length;i++)
07              System.out.print(intarray[i]+"   ");
08          int temp;
09          System.out.println();
10          for(int i=0;i<intarray.length;i++){
11
12              for(int j=i;j<intarray.length;j++){
13                  if(intarray[j]<intarray[i]){
14                      temp=intarray[i];
15                      intarray[i]=intarray[j];
16                      intarray[j]=temp;
17                  }
18              }
19
20          }
21          System.out.println("排序后的数组");
22          for(int i=0;i<intarray.length;i++){
23              System.out.print(intarray[i]+"   ");
24          }
25      }
26  }
```

【运行结果】程序运行结果如下。

```
排序前的数组：
1   2   33   5   34   67   213   8
排序后的数组
1   2   5   8   33   34   67   213
```

【代码分析】在程序中首先声明了一个数组，并输出其内容，然后对数组进行冒泡排序，最后输出排序后的数组内容。

3.3 多维数组

多维数组是指使用多个索引来访问数组的元素，它适用于表示表或者其他比较复杂的内容。声明一个多维数组，需要一组括号来制定它的下标，如下所示。

```
int[][] Two=new int[5][5];
```

数组声明完成后，就需要对数组进行赋值，如下所示。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
Two={
{1,2,3,4,5},
{6,7,8,9,0},
{11,12,13,14,15},
{16,17,18,19,20},
{21,22,23,24,25}
}
```

对多维数组赋值也可以使用 for 循环进行。

```
for(int i=0;i<Two.length;i++){
for(int j=0;j<Two[i].length;j++)
Two[i][k]=k++;
}
```

上面的格式中，第一个循环中的 Two.length 表示的是该多维数组的行数，而第二个循环中的 Two[i].length 则表示的是该多维数组的列数。

【范例 3-7】示例代码 3-7 是一个测试多维数组的程序。

示例代码 3-7

```
01  public class TwoTest {
02      public static void main(String[] args) {
03          int[][] two1={
04                   {1,2,3,4,5},
05                   {6,7,8,9,10},
06                   {11,12,13,14,15},
07                   {16,17,18,19,20},
08                   {21,22,23,24,25}
09                   };
10          //第二种创建方式
11          int[][] Two=new int[5][5];
12          int k=1;
13          for(int i=0;i<Two.length;i++){
14              for(int j=0;j<Two[i].length;j++)
15              Two[i][j]=k++;
16              }
17          System.out.println("输出 two1 数组");
18          for(int i=0;i<two1.length;i++){
19              for(int j=0;j<two1[i].length;j++){
20                  System.out.print(two1[i][j]+"  ");
21                              }
22              System.out.println();
23
24          }
25          System.out.println("输出 Two 数组");
26          for(int i=0;i<Two.length;i++){
27              for(int j=0;j<Two[i].length;j++){
28                  System.out.print(Two[i][j]+"  ");
29                              }
30              System.out.println();
31
32          }
33      }
34  }
```

【运行结果】程序运行结果如下。

```
输出 two1 数组
1  2  3  4  5
6  7  8  9  10
11  12  13  14  15
16  17  18  19  20
21  22  23  24  25
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
输出 Two 数组
1  2  3  4  5
6  7  8  9  10
11  12  13  14  15
16  17  18  19  20
21  22  23  24  25
```

以上程序中分别使用了两种不同的方法创建多维数组，并将两个多维数组打印输出。在 Java 中的多维数组实质上都是一维数组。若每一个数组的元素数不同，就会形成不规则数组。例如，声明一个不规则数组的格式如下。

```
int[][] two=new int[4][];
two[0]=new int[1];
two[1]=new int[2];
two[2]=new int[3];
two[3]=new int[4];
```

在上面的声明中，第一行有一个元素，第二行有两个元素，第三行有三个元素，第四行有四个元素。

【范例 3-8】示例代码 3-8 是一个测试不规则数组的程序。

示例代码 3-8

```
01  public class Two {
02      public static void main(String[] args) {
03
04          int[][] two=new int[4][];
05          two[0]=new int[1];
06          two[1]=new int[2];
07          two[2]=new int[3];
08          two[3]=new int[4];
09
10          int k=1;
11
12          for(int i=0;i<two.length;i++){
13              for(int j=0;j<two[i].length;j++)
14                  two[i][j]=k++;
15          }
16              System.out.println("所得到的不规则数组为：");
17              for(int i=0;i<two.length;i++){
18                  for(int j=0;j<two[i].length;j++)
19                      System.out.print(two[i][j]+"   ");
20                  System.out.println();
21              }
22      }
23  }
```

【运行结果】程序的运行结果如下。

```
所得到的不规则数组为：
1
2   3
4   5   6
7   8   9   10
```

在程序中，two.length 表示的是数组 two 的行数，而 two[i].length 表示的是第 i 行的列数。

3.4 习题

【习题 3-1】首先声明一个数组，将该数组中存在的重复元素删除，并将该数组按照一定的

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



顺序排序。

【分析】该题主要考查的是数组中的排序操作，数组的排序方法主要有冒泡排序法和选择排序法两种。

【核心代码】

```
for (int i = 0; i < array.length; i++) {// 利用随机数随意产生 20 个 0～20 之间的随机数
            array[i] = rd.nextInt(20);// 给 array 数组赋值
        }
        Arrays.sort(array);                    // 对 array 数组进行升序排序
        for (int i = 0; i < array.length; i++) {// 将 array 数组中的数据输出
            System.out.print(" " + array[i]);
            if ((i + 1) % 5 == 0)
                System.out.println();
        }
        int c[] = trim(array);                 // 调用 trim 方法去掉重复数字
        System.out.println("\n 去掉数组中重复数字,还剩有" + c.length + "个数字,它们的
值分别为：");
        for (int i = 0; i < c.length; i++) { // 将去掉重复数字的数组元素输出
            System.out.print(c[i] + " ");
        }
```

【习题 3-2】定义一个双精度类型的数组，使用 for 循环对每一个元素进行赋值。并计算出该数组的平均值，以及最大值和最小值。

【分析】该题主要考查了数组的初始化操作，首先使用 for 循环对数组中的每个元素进行赋值，然后使用 for 循环结合条件运算符将最大值与最小值求出。要求平均值，只要将数组元素的和值算出，与它的长度相除就可以得到了。

【核心代码】

```
public static double[] show(double[] dou) { // 求数组元素中的最大值、
                                             // 最小值和所有元素的平均值
        double[] result = new double[3];    // 创建一个长度为 3 的 double 型数组
        double max = dou[0], min = dou[0], sum = dou[0];
        for (int i = 1; i < dou.length; i++) {
            max = max > dou[i] ? max : dou[i];       // 求最大值
            min = min < dou[i] ? min : dou[i]  ;     // 求最小值
            sum += dou[i];                           // 求所有元素的累加和
        }
        result[0] = max;
        result[1] = min;
        result[2] = sum / dou.length;                // 求平均值
        return result;
    }
}
```

【习题 3-3】使用多维数组打印一个奇数阶幻方，也就是由一个行数和列数都相等的数据组成一个方阵，即由 1～n×n 个数据组成，由于是奇数阶幻方，所以阶数 n 必须是奇数。并且要求每行、每列和对角线上的数据之和相等。

【分析】奇数阶幻方指的是由一个行数和列数都相等的数据组成一个方阵，即由 1～n×n 个数据组成，由于是奇数阶幻方，所以阶数 n 必须是奇数。并且要求每行、每列和对角线上的数据之和相等。

【核心代码】

```
int a[][] = new int[n][n];
int i=n-1, j=n/2;
for (int k = 1; k <= n * n; k++) {
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
        a[i++][j++] = k;
        if (k % n == 0) {
            i -= 2;
            j--;
        } else {
            i = i%n;
            j = j%n;
        }
    }
```

【习题 3-4】求出第 3 题中幻方的两条对角线元素值之和（交叉位置的元素只计算一次），将它们的和值输出到终端上。

【分析】由于 n 阶幻方两条对角线上的数字之和相等，所以将一条对角线数据之和乘以 2，再减去交叉位置的数字即可。

【核心代码】

```
int z=0, k=0, m=0;
   for (int x = 0; x < a.length; x++) {
      for (int y = 0; y < a[x].length; y++) {
         k = a[x][y];
         if (x == y) {
            z += k;
            if (x * 2 + 1 == n) {
               m = k;
            }
         }
      }
   }
```

【习题 3-5】使用数组循环初始化的方法来创建一个数组。

【分析】首先声明一个数组，遍历数组，为每一个元素赋值。

【核心代码】

```
int[] array = new int[5];
for (int i = 0; i < array.length; i++) {
   array[i] = new Random().nextInt();
}
```

【习题 3-6】将第 5 题所创建的数组复制，并对其进行排序。

【分析】新建一个数组引用指向第 5 题的数组，然后用冒泡法进行排序。

【核心代码】

```
int tmp;
for (int i = 0; i < array1.length; i++) {
   for (int j = i; j < array1.length; j++) {
      if (array1[j] < array1[i]) {
         tmp = array1[i];
         array1[i]=array1[j];
         array1[j]=tmp;
      }
   }
}
```

【习题 3-7】在第 6 题复制的数组中，随机查找几个元素。

【分析】因为第 6 题中数组的长度为 4，所以随机产生几个上限为 4 的数组索引，便可以进行随机查找。

【核心代码】

```
Random r = new Random();
int i = r.nextInt(4);
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
System.out.println("随机查找："+ array1[i]);
i = r.nextInt(4);
System.out.println("随机查找："+ array1[i]);
i = r.nextInt(4);
System.out.println("随机查找："+ array1[i]);
```

【习题 3-8】声明一个整型数组，将其转换为一个字符串数组。

【分析】声明一个整型数组，遍历此数组，将每个元素转换为字符串类型后，赋值给同样大小的字符串数组的相应元素。

【核心代码】

```
public class ConvIntArray {
   public static void main(String[] args) {
      Integer[] is = {49, 72, 86, 45, 66, 59};
      String[] ss = new String[is.length];
      for (Integer i : is) {
         ss[i] = i.toString();
      }
   }
}
```

【习题 3-9】将一个数组的所有元素连接成一个字符串。

【分析】声明一个数组，遍历数组，将所有元素连接为一个字符串。

【核心代码】

```
public class RemoveDupl {
   public static void main(String[] args) {
      String s = "";
      String[] ss = {
            "劝君莫惜金缕衣，",
            "劝君惜取少年时。",
            "花开堪折直须折，",
            "莫待无花空折枝。"
      };
      for (String a : ss) {
         s += a;
      }
      System.out.println(s);
   }
}
```

【习题 3-10】将一个整型数组的索引为偶数的元素变为原来的两倍。

【分析】声明一个数组，遍历数组元素，如果元素索引为偶数，则将其值变为原来的两倍。

【核心代码】

```
public class EvenNumberIndex {
   public static void main(String[] args) {
      Integer[] is = {17, 29, 15, 26, 33, 51};
      for (int i = 0; i < is.length; i++) {
         if ((i % 2) == 0) {
            is[i] = is[i]*2;
         }
      }
      for (int i : is) {
         System.out.println(i);
      }
   }
}
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第4章 字 符 串

字符串不属于基本数据类型，但是在程序的编写中也会经常使用到。本章主要介绍字符串的使用。



4.1 字符串

拿出一张白纸，然后在纸上写下一句话，这句话就相当于一个字符串。字符串是由单个或者多个字符所组成的。字符串也相当于 char 类型的数组，字符串用 String 来表示，并且在该类中有一系列的属性和方法，提供对字符串的各种操作的支持。

Java 中提供了两种字符串类，分别是 String 类和 StringBuffer 类，并且为它们提供了各自相应的实现字符串的方法。

4.1.1 声明字符串

声明字符串就好比是去超市买东西，进门的时候，要先拿一个菜篮子或者一个小推车，以此表明我要准备买东西，然后就需要往篮子或推车内放入需要购买的东西。声明一个字符串有两种方式，即常量声明方式和对象声明方式。

- 常量声明方式是采用双引号括住一个字符串的方法，示例如下。

```
String hello= “HelloWorld”;
```

- 对象声明方式的格式如下。

```
字符串类 字符串名 =New 字符串类（参数）;
String hello=New String(“HelloWorld”);
```

4.1.2 创建字符串

进入超市并已经推了推车，接下来就需要将需要购买的物品放置到推车中，也就是创建字符串。

String 类字符串的创建有几种方式。

- `String str=new String();`

初始化一个新创建的 String 对象，该对象表示一个空字符串。

- `String str=new String(“String”);`

初始化一个新创建的 String 对象，该对象表示参数所表示的字符串。

- `String str=new String (char char[]);`

初始化一个新创建的 String 对象，该对象表示字符数组顺序组成的字符串。

- `String str=new (char char[],int index,int num);`

初始化一个新创建的 String 对象，该对象表示由字符数组的子字符数组组成的字符串，代码中的第二个参数表示起始位置，第三个参数表示字符的数量。

- `String str=new String(int[],int begin,int end);`

初始化一个新创建的 String 对象，该对象表示由 Unicode code point 数组的一个子数组所

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

组成的字符串。

- `String str=new String(byte[] bytes)`

初始化一个新创建的 String 对象，该对象通过使用系统默认的字符集解码 byte 数组，构造一个新的对象。

- `String str=new String(byte[] bytes, int offset, int length)`

初始化一个新创建的 String 对象，该对象通过使用系统默认的字符集解码 byte 数组，构造一个新的对象，第二个参数表示其实位置，第三个参数表示字符数量。

- `String str=new String(byte[] bytes, int offset, int length, String charsetName)`

初始化一个新创建的 String 对象，该对象通过使用指定的 charset 解码指定的 byte 数组，构造一个新的对象，第二个参数表示其实位置，第三个参数表示字符数量。

- `String str=new String(byte[] bytes, String charsetName)`

初始化一个新创建的 String 对象，该对象通过使用指定的 charset 解码指定的 byte 数组，构造一个新的对象。

- `String str=new String(StringBuffer buffer)`

初始化一个新创建的 String 对象，该对象表示由 StringBuffer 类型参数所包含的字符序列。

4.2　字符串操作

本节主要介绍字符串的操作，其中主要包括有字符串的连接与复制、更改字符串的内容、获取字符串的长度等。

4.2.1　字符串的比较

有时候需要对字符串进行比较，比较其中内容是否相同。这好比张三、李四、王五、赵六都在看书，这时并不知道书名，只知道他们看的内容，若要判断他们所看的是否是同样的一本书，就需要对其内容进行对比。

对于字符串内容的比较，主要有以下几个方法。

- equals()：比较两个字符串的内容是否相等，返回值为 Boolean 类型。
- equalsIgoreCase()：忽略大小写，比较两个字符串的内容是否相等，返回值为 Boolean 类型。
- compareTo(String anotherString)：按照词典的顺序进行比较，返回 int 类型值。

注意：在比较字符串时，不能使用“==”，因为使用“==”比较对象时，实际上判断的是是否为同一个对象，如果内容相同，但不是同一个对象，其返回值也是 false，也即其对应的不是同一个内存地址。

【范例 4-1】示例代码 4-1 是一个测试字符串比较的程序。

示例代码 4-1

```
01  public class Bijiao {
02
03      public static void main(String[] args) {
04
05          String zhagnsan="Think In Java!";
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
06          String lisi="Think In Java!";
07          String wangwu="think in java";
08          String zhaoliu="Think in Java";
09
10          if(zhagnsan.equals(lisi))
11              System.out.println("张三和李四读书内容相同");
12          else
13              System.out.println("张三和李四读书内容不同");
14
15          if(lisi.equals(wangwu))
16              System.out.println("李四和王五读书内容相同");
17          else
18              System.out.println("李四和王五读书内容不同");
19
20          if((wangwu.compareTo(zhaoliu))>0)
21              System.out.println("王五和赵六读书内容相同");
22          else
23              System.out.println("王五和赵六读书内容不同");
24      }
25  }
```

【运行结果】程序运行结果如下。

```
张三和李四读书内容相同
李四和王五读书内容不同
王五和赵六读书内容相同
```

【代码解析】在以上的程序中，定义了三个字符串变量，并为三个变量赋予了初始值，然后分别使用 equals、equalsIgnoreCase、compareTo 方法对三个字符串变量进行内容的判断，并输出判断结果。

4.2.2 连接字符串

当一个物体从中间折断，或者是两个物体需要合并成一个物体时，我们会想办法将二者粘起来。在字符串中，也可以将两个不同的字符串连接起来。将字符串进行连接的方法是 concat()，该方法可以实现将两个或者多个字符串连接为一个字符串，这个方法的参数是一个 String 对象，返回值也是一个 String 对象。

【范例 4-2】示例代码 4-2 是一个测试字符串连接的程序。

示例代码 4-2

```
01  public class Lianjie {
02      public static void main(String[] args) {
03          String str="Hello ";
04          String s="World ";
05          String b="!";
06
07          String a=str.concat(s);
08          String c=str.concat(str).concat(b);
09
10          System.out.println(a);
11          System.out.println(c);
12
13      }
14  }
```

【运行结果】程序运行结果如下。

```
Hello World
Hello Hello !
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



【代码解析】在程序中，首先定义了三个字符串对象，接着使用 concat()方法将前两者进行连接，然后连续使用两次 concat()方法，将三个字符串进行连接，最后打印输出，将二者进行对比。

4.2.3　复制字符串

在复制文稿时，通常使用复印机，只需要将文稿放到复印机中，按下复印按钮，就会有相同的文稿从复印机中传出。对于字符串，也有类似的复制方法。字符串的复制，主要由方法 copyValueOf()来实现，其中提供有两种参数方式，分别完成对字符串的全部复制和部分复制。

```
String copyValueOf(char[],int offset,int count)
```

这个方法是将字符数组的一部分内容作为一个新的字符串。

```
String copyValueOf(char[])
```

该方法是复制整个字符数组作为一个新的字符串。

【范例 4-3】示例代码 4-3 是一个测试字符串复制的程序。

示例代码 4-3

```
01  public class Copy {
02      public static void main(String[] args) {
03
04          String str="HelloWorld!";
05
06          char[] ch=new char[str.length()];
07          for(int i=0;i<str.length();i++)
08              ch[i]=str.charAt(i);
09
10          String s=String.copyValueOf(ch);
11          String st=String.copyValueOf(ch,0,5);
12
13          System.out.println(s);
14          System.out.println(st);
15
16      }
17  }
```

【程序结果】程序运行结果如下。

```
HelloWorld!
Hello
```

【代码解析】在程序中，首先创建一个字符串，内容为“HelloWorld”，字符串名为 str，然后创建一个 char 类型数组 ch，并设置数组长度为字符串的长度，并将字符串 str 转化为 char 数组并赋值给 ch，使用 copyValueOf()方法复制 ch 的内容，分别赋值全部内容和复制五个字符，最后输出结果。

4.2.4　更改字符串的内容

若发现写到纸上的字错了，如果使用的是铅笔，则可以用橡皮擦掉，如果使用的是钢笔，则可以使用修正液来进行修改。对于字符串，也提供了修改的方法。

更改字符串内容的方法，也就是替换方法，共有三种，如下所示。

- replace(char,char)：该方法可以把字符串中与方法的第一个参数字符相同的字符统一替换为方法中的第二个字符。
- replaceAll(String ,String)：该方法可以将字符串中与方法的第一个参数字符串相同的字

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

符串统一替换成方法中的第二个字符串。

- replaceFirst(String ,String)：该方法可以将字符串中与第一个参数字符串相同的第一个字符串替换为方法中的第二个字符串。

【范例 4-4】示例代码 4-4 是一个测试字符串更改的程序。

示例代码 4-4

```
01  public class Xiugai {
02      public static void main(String[] args) {
03
04          String s1=new String("第一个测试");
05          String str=s1.replace('一', '二');
06          System.out.println(str);
07
08          String s2=new String("第二个测试");
09          String str2=s2.replaceAll("二", "三");
10          System.out.println(str2);
11
12          String s3=new String("第三个测试");
13          String str3=s3.replaceFirst("三", "四");
14          System.out.println(str3);
15      }
16  }
```

【运行结果】程序运行结果如下。

```
第二个测试
第三个测试
第四个测试
```

【代码解析】在以上程序中，首先定义了三个字符串，然后分别用前面所介绍到的三种方法对字符串内容进行修改，最后对结果打印输出并进行比对。

4.2.5 获取字符串的长度

在体育课上，老师经常让第一排的同学报数，目的是要获得第一排的人数，也就是第一排的长度。在字符串中提供了一个获取字符串长度的方法。

String 类对象的字符串长度值是固定的，一旦字符串对象创建以后，其长度就不会再改变。在 String 类中提供了一个 length()方法，用于获取字符串的长度。

【范例 4-5】示例代码 4-5 是一个测试字符串长度的程序。

示例代码 4-5

```
01  public class Length {
02      public static void main(String[] args) {
03
04          String str="第一队列";
05          String st="第二队列";
06
07          int i=str.length();
08          int j=st.length();
09
10          System.out.println("第一队列的长度是："+i);
11          System.out.println("第二队列的长度是："+j);
12
13      }
14  }
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

【运行结果】程序运行结果如下。

```
第一队列的长度是：4
第二队列的长度是：4
```

【代码解析】在以上程序中，首先定义了两个字符串对象，然后使用 length()方法获取了字符串的长度，并赋值给整型变量，最后打印输出结果。

4.2.6　分割字符串

夏天吃西瓜时，首先需要将西瓜切成瓣，这样方便一块一块地拿着吃。对于字符串，也有专门的方法将其切割。

分割字符串使用的方法是split()，首先要按照指定的划界表达式将字符串分割成几部分，每一部分是一个字符串，方法的返回值是一个字符串数组。

【范例 4-6】示例代码 4-6 是一个测试分割字符串的程序。

示例代码 4-6

```
01  public class Qiege {
02      public static void main(String[] args) {
03
04          String str="张三，回家吃饭，快点";
05
06          String[]st=str.split("，");
07          System.out.println("原句是："+str);
08
09          for(int i=0;i<st.length;i++){
10
11              System.out.println("剪切后的第"+(i+1)+"部分是："+st[i]);
12
13          }
14      }
15  }
```

【运行结果】程序运行结果如下。

```
原句是：张三，回家吃饭，快点
剪切后的第 1 部分是：张三
剪切后的第 2 部分是：回家吃饭
剪切后的第 3 部分是：快点
```

【代码解析】在以上程序中，首先定义了一个字符串，然后使用 split()方法将字符串按照“，”来进行分割，最后输出原句和分割后的字符串数组，进行比较。

4.3　StringBuffer

StringBuffer 和 String 不同，String 对象的长度是不可以改变的，而 StringBuffer 的长度是可以改变的，如果使用 String 频繁地去进行字符串的拼接操作，则会产生很多的对象。若使用 StringBuffer 类，则可以解决这些问题。

StringBuffer 类中的一些常用方法如下。

- length()：返回字符串的长度。
- insert()：在字符串的中间插入，有几种不同的参数方式。
- append()：在字符串的末尾追加字符串。
- setCharAt(int index,char ch)：设置指定索引位置为制定的字符。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

- toString()：返回字符串。

【范例 4-7】示例代码 4-7 是一个测试 StringBuffer 的程序。

示例代码 4-7

```
01  public class StringBufferTest {
02
03      public static void main(String[] args) {
04
05          StringBuffer sb=new StringBuffer();
06          sb.append("张三，");
07          sb.append("回家，");
08          sb.append("吃饭。");
09
10          System.out.println(sb.toString());
11
12          int i=sb.length();
13          System.out.println("StringBuffer 的长度是："+i);
14
15
16      }
17
18  }
```

【运行结果】程序的运行结果如下。

```
张三，回家，吃饭。
StringBuffer 的长度是：9
```

【代码解析】在以上程序中，首先定义了一个 StringBuffer 对象，然后使用 append()方法为其追加内容，并使用 toString()方法将所有追加的内容打印输出，最后使用 length()方法来返回 StringBuffer 的长度，并打印输出。

对于 String 类和 StringBuffer 类，二者有以下不同。

- String 类：该类一旦产生一个字符串，其对象就不可以改变，并且字符串的内容和长度是固定不变的。如果在程序中需要调用该字符串的信息，就需要调用系统所提供的各种字符串操作方法。通过这些方法对字符串进行操作，并不会改变对象实例本身，而是产生了一个新的对象实例。并且系统在为 String 类对象分配内存时，也是按照对象所包含的实际字符数来分配的。
- StringBuffer 类：该类具有缓冲的功能。StringBuffer 类处理可变字符串。如果要修改一个 StringBuffer 类的字符串，不需要再创建新的字符串对象，可以直接操作原来的字符串。该类中有些方法与 String 类不同。系统在为 StringBuffer 类对象分配内存时，除去当前字节所占的空间外，还另外提供了 16 个字符大小的缓冲区。Length 方法可以返回当前实际包含字符串的长度，而 capacity 方法则可以返回当前数据容量和缓冲区的容量之和。

4.4 习题

【习题 4-1】分别用常量声明、对象声明和字符数组三种方式创建字符串。

【分析】此题考查字符串的创建方式。

【主要代码】

```
String s1 = "This is a String.";
String s2 = new String("This is another String.");
char[] cs = {'a', ' ', 'S', 't', 'r', 'i', 'n', 'g'};
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
String s3 = new String(cs);
```

【习题 4-2】声明几个字符串，分别比较字符串的内容。

【分析】此题考查字符串的三种比较方式。

【主要代码】

```
String s1 = "This is a String.";
String s2 = "This is a string.";
String s3 = "this is a String.";
String s4 = "this is a string.";

System.out.println("s1 与 s2 是否一样："+ (s1.equals(s2)));
System.out.println("s1 与 s3 是否一样："+ (s1.equalsIgnoreCase(s3)));
System.out.println("s1 与 s4 是否一样："+ (s1.compareTo(s4) > 0));
```

【习题 4-3】声明几个字符串，将其连接成一个字符串。

【分析】此题考查字符串的连接。

【主要代码】

```
String s1 = "This ";
String s2 = "is ";
String s3 = "a ";
String s4 = "String.";

String s = s1.concat(s2).concat(s3).concat(s4);
```

【习题 4-4】声明一个字符数组，复制其部分或全部，构造一个字符串。

【分析】此题考查字符串的复制。

【主要代码】

```
char[] cs = {'T', 'h', 'i', 's', ' ', 'i', 's', ' ', 'a', ' ', 'S', 't', 'r', 'i',
'n', 'g'};
String s1 = String.copyValueOf(cs);
String s2 = String.copyValueOf(cs, 8, 8);
```

【习题 4-5】改正字符串中的错误字符。

【分析】此题考查字符串的替换。

【主要代码】

```
String s = "This is a String.";

String s1 = s.replace('S', 's');
String s2 = s.replace("This", "That");
String s3 = s.replaceAll("\\s+", "_");
```

【习题 4-6】计算字符串长度。

【分析】此题考查字符串的 length()方法的使用。

【主要代码】

```
String s1 = "字符串长度";
String s2 = "字符串长度测试";

int a = s1.length();
int b = s2.length();
```

【习题 4-7】声明一个字符串，按照特定的符号将其分为几部分。

【分析】此题考查字符串的分隔。

【主要代码】

```
String s = "This is a String.";
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
String[] ss = s.split("\\s+");
```

【习题 4-8】声明几个字符串，使用 StringBuffer 将其连接为同一个字符串。

【分析】此题考查 StringBuffer 的使用。

【主要代码】

```
String s1 = "This ";
String s2 = "is ";
String s3 = "a ";
String s4 = "String.";

StringBuffer sb = new StringBuffer();
sb.append(s1).append(s2).append(s3).append(s4);
```

【习题 4-9】有一个字符串“Love makes man grow up.”，现在要求在“up”后加上“ or sink down.”。

【分析】此题考查 StringBuffer 的使用。

【主要代码】

```
public class InsertString {
   public static void main(String[] args) {
      String s = "Love makes man grow up or sink down. ";
      String s1 = "Love makes man grow up. ";
      String s2 = "or sink down. ";

      StringBuffer sb = new StringBuffer();
      sb.append(s1);
      sb.insert(sb.length() - 1, s2);

      System.out.println(sb);
   }
}
```

【习题 4-10】使用 StringBuffer 替换字符串中的某些部分。

【分析】此题考查 StringBuffer 的使用。

【主要代码】

```
public class ReplaceByStringBuffer {
   public static void main(String[] args) {
      StringBuffer sb = new StringBuffer("玲珑骰子安红豆，开不完春柳春花满画楼。");
      sb.replace(sb.indexOf("，") + 1, sb.length() - 1, "入骨相思知不知？");
      System.out.println(sb);
   }
}
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第 5 章　程序控制语句

一个人在做事之前，总是要经过逻辑思维的判断，在程序中也是如此。Java 语言中提供了判断分支语句、多分支语句、switch 语句、循环语句和跳转语句等，可以使程序既能够按照自上而下的顺序执行，又能够在遇到某些条件进行跳转，还可以对某些操作进行重复执行。

本章主要介绍 Java 的流程控制语句，通过本章的学习，希望读者可以掌握各种流程控制语句，包括选择结构 if/switch 语句、循环结构的 for/while 语句，并且可以综合运用这些流程控制语句。



5.1　顺序结构

在银行中办理业务，首先要取一个办理业务的号码，然后银行会依照号码按顺序一个一个地进行办理，其中不允许插队，直到完成，停止业务办理。

顺序结构是程序中最常见的流程结构，语句可以按照出现的顺序来执行，其中没有跳转或者判断，直到程序的结束为止。

【范例 5-1】示例代码 5-1 是一个测试顺序结构的程序。

示例代码 5-1

```
01  public class OrderTest {
02      public static void main(String[] args) {
03          //声明变量
04          int x;
05          int y;
06          int z;
07          //为变量赋值
08          x=1;
09          y=2;
10          z=x+y;
11          //将变量的值打印输出
12          System.out.println("x 的值是："+x);
13          System.out.println("y 的值是："+y);
14          System.out.println("z 的值是："+z);
15
16      }
17  }
```

【运行结果】运行结果如下。

```
x 的值是：1
y 的值是：2
z 的值是：3
```

【代码解析】在程序中包含有三条赋值的语句，是按照自上而下的顺序来执行的。



5.2　条件语句

选择结构也是分支结构，其中包括条件判断语句，该语句可以根据一个表达式的判断结果

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

来选择不同的分支执行语句，可以根据不同的条件来执行不同的动作。条件判断语句包括 if 语句和 switch 语句。

5.2.1 if 语句

有时候，我们对待一件事情，总是会想，如果事情向这个方向发展的话，我们该怎么做，如果向另一个方向发展，我们又该怎么去做。在程序中，也有类似的语句，就是 if 语句，用于条件判断。

if 语句是最简单的条件判断语句。if 语句的基本语法格式如下。

```
if(布尔表达式){
    //语句 1
}
    //语句 2
```

在该语句中，if 是关键字，表示此语句是 if 条件语句。其中有一个返回值为布尔类型值的表达式，要根据该表达式的结果来判断选择分支。如果表达式的结果返回为 true，则执行处理语句。如果在 if 条件语句中只有一条执行结果，可以省略掉“{}”。if 语句结构图如图 5-1 所示。

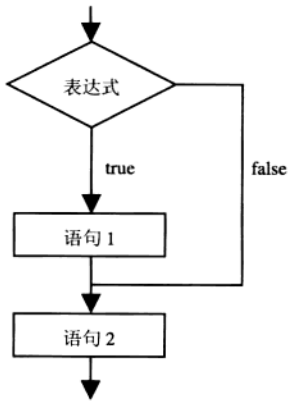


图 5-1 if 语句结构图

【范例 5-2】示例代码 5-2 是一个测试 if 语句的程序。

示例代码 5-2

```
01  public class IfTest {
02      public static void main(String[] args) {
03
04          int zage=22;
05          int lage=23;
06
07          if(zage>lage){
08              System.out.println("张三年龄是："+zage+"岁");
09              System.out.println("李四年龄是："+lage+"岁");
10              System.out.println("张三比李四大");
11          }
12          if(lage>zage){
13              System.out.println("张三年龄是："+zage+"岁");
14              System.out.println("李四年龄是："+lage+"岁");
15              System.out.println("李四比张三大");
16          }
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
17
18        }
19
20    }
```

【运行结果】程序运行结果如下。

```
张三年龄是：22 岁
李四年龄是：23 岁
李四比张三大
```

【代码解析】在以上的代码中，首先定义了两个整型变量 zage 和 lage，作为张三和李四的年龄，然后使用 if 语句进行判断，如果 zage 值大于 lage，则输出“张三比李四大”，反之则输出“李四比张三大”。

if 条件语句中的另一种形式是 if-else，if-else 的形式要比 if 的形式复杂一些。多出了 else 的部分，else 部分是在 if 语句的布尔表达式的值为 false 时执行的语句，其格式如下。

```
if (布尔表达式) {
            //处理语句 1
        } else {
            //处理语句 2
        }
```

在该条件语句天中，首先判断布尔表达式的值，如果布尔表达式的值是“true”，则会去执行处理语句 1，如果布尔表达式的值是“false”，则会去执行处理语句 2。图 5-2 是 if-else 语句的流程图。

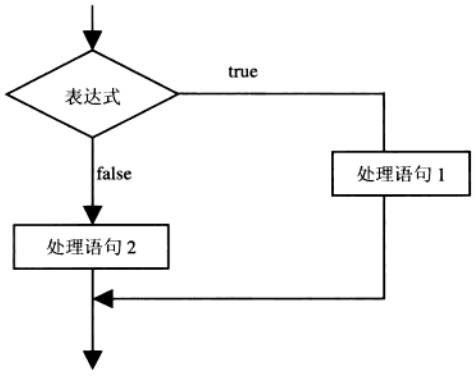


图 5-2　if-else 语句流程图

【范例 5-3】示例代码 5-3 是一个测试 if-else 语句的程序。

示例代码 5-3

```
01  public class Ifelse {
02      public static void main(String[] args) {
03
04          int zhagnsan=90;
05          int lisi=95;
06
07          if(zhagnsan>lisi){
08              System.out.println("张三赢了！");
09          }else{
10              System.out.println("李四赢了！");
11          }
12
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
13        }
14    }
```

【运行结果】程序运行结果如下。

```
李四赢了!
```

【代码解析】在以上代码中，为两个整型变量 zhangsan 和 lisi 分别赋初始值，使用 if-else 进行判断，如果 zhagnsan 大于 lisi，则输出“张三赢了!”，反之则输出“李四赢了!”。

if 语句中的第三种形式是 if-else if，是 if 语句中最为复杂的一种，可以看做是多个 if-else 进行拼接，其格式如下。

```
if (布尔表达式 1) {
    处理语句 1;
    } else if(布尔表达式 2){
    处理语句 2;
    }else if(布尔表达式 3){
    处理语句 3;
    }
```

这种表达形式是当“布尔表达式 1”的值是“true”时，执行处理语句 1，如果为“false”则去判断“布尔表达式 2”，如果“布尔表达式 2”值为“true”，则执行处理语句 2，否则判断“布尔表达式 3”，如果“布尔表达式 3”的值为“true”，则执行处理语句 3。图 5-3 是 if-else if 语句的流程图。

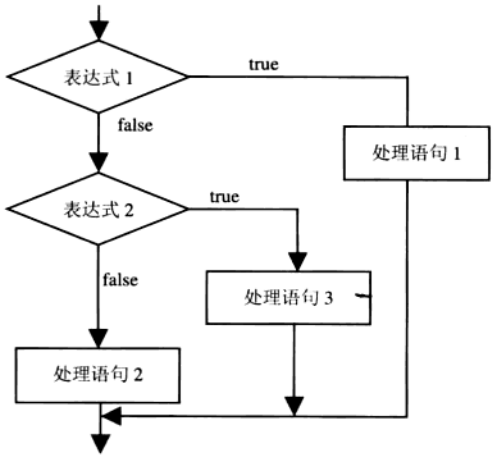


图 5-3　if-else if 语句流程图

【范例 5-4】示例代码 5-4 是一个测试 if-else if 语句的程序。

示例代码 5-4

```
01  public class IfElseTest {
02      public static void main(String[] args) {
03
04          int i=88;
05          if(i>90){
06              System.out.println("张三的成绩是："+i+"\n"+"评价是：非常优秀！");
07          }else
08          if(i>80){
09              System.out.println("张三的成绩是："+i+"\n"+"评价是：优秀！");
10          }else
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
11            if(i>70){
12                System.out.println("张三的成绩是："+i+"\n"+"评价是：良好！");
13            }else
14            if(i>60){
15                System.out.println("张三的成绩是："+i+"\n"+"评价是：及格");
16            }else
17            {
18            System.out.println("张三的成绩是："+i+"\n"+"评价是：不及格，该努力了!!");
19            }
20
21        }
22    }
```

【运行结果】程序运行结果如下。

```
张三的成绩是：88
评价是：优秀!
```

【代码解析】在以上代码中，首先定义了一个整型变量，作为张三的成绩，然后使用 if-else if 语句进行判断，并输出张三的成绩和成绩段的评价。

5.2.2　switch 语句

在 Java 中，存在 switch 语句，属于多分支结构，可以根据表达式的值来选择一个分支执行。

switch 语句取代了比较复杂的 if-else if 语句，并且其表达式的值只能是 byte、short、int 或 char 类型。switch 语句的语法格式如下。

```
switch (表达式) {
        case value1:
            处理语句1;
            break;
        case value2:
            处理语句2;
            break;
        case value3:
            处理语句3;
            break;
        ……
    }
```

在 switch 语句中，首先会去执行表达式，然后使用表达式的值和各分支的值进行对比，如果和 value1 的值相同，就会执行处理语句 1，如果等于 value2，就会执行处理语句 2，等等，在执行完毕后，执行 break 跳出。图 5-4 是 switch 语句的流程图。

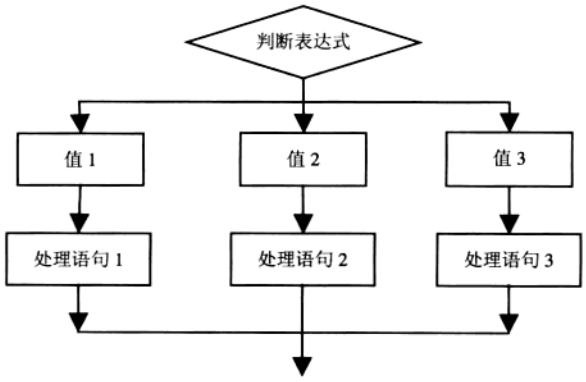


图 5-4　switch 语句流程图

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

【**范例 5-5**】示例代码 5-5 是一个测试 switch 语句的程序。

示例代码 5-5

```
01  public class SwitchTest {
02      public static void main(String[] args) {
03          int i=1;
04
05          switch (i) {
06          case 1:
07              System.out.println("今天要去数学辅导班");
08              break;
09          case 2:
10              System.out.println("今天要去语文辅导班");
11              break;
12          case 3:
13              System.out.println("今天要去英语辅导班");
14              break;
15          case 4:
16              System.out.println("今天要去物理辅导班");
17              break;
18          case 5:
19              System.out.println("今天要去化学辅导班");
20              break;
21          case 6:
22              System.out.println("今天要去生物辅导班");
23              break;
24          case 7:
25              System.out.println("今天要去政治辅导班");
26              break;
27
28          }
29
30      }
31  }
```

【运行结果】程序运行结果如下。

```
今天要去数学辅导班
```

【代码解析】以上程序中，首先定义了一个整型变量，并将变量赋值为 1，然后使用 switch 语句进行判断，并输出相应的语句。



5.3 循环语句

在程序中，有时候需要对某一段代码进行反复的操作，这时就需要使用循环语句，包括 for 循环和 while 循环。

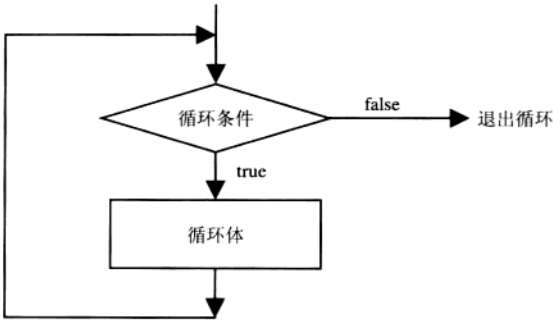
5.3.1 for 循环

循环语句可以使计算机重复地执行某一段代码，同时也可以通过指定的终止条件来控制程序的无限循环。for 循环是最常用的一个循环语句，其语法如下。

```
For(表达式 1;可返回布尔值的表达式;表达式 2){
//相关语句
}
```

for 循环可以根据可返回布尔值的表达式来决定是否执行相关的语句。当返回值为“true”

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



时，for 语句就会执行相关语句并且会再次判断可返回布尔值的表达式，如果该表达式的值仍然是“true”，则再次执行相关语句，直到返回值为“false”为止。图 5-5 是 for 循环的流程图。

图 5-5　for 循环流程图

【范例 5-6】示例代码 5-6 是一个测试 for 循环语句的程序。

示例代码 5-6

```
01  public class ForTest {
02      public static void main(String[] args) {
03
04          for(int i=0;i<3;i++){
05
06              System.out.println("外层第"+i+"次循环");
07              for(int j=0;j<3;j++){
08                  System.out.println("内层第"+j+"次循环");
09
10                  int k=5;
11              }
12              System.out.println();
13
14          }
15
16      }
17  }
```

【运行结果】运行结果如下。

```
这是外面第 0 次循环
这是里面第 0 次循环
这是里面第 1 次循环
这是里面第 2 次循环

这是外面第 1 次循环
这是里面第 0 次循环
这是里面第 1 次循环
这是里面第 2 次循环

这是外面第 2 次循环
这是里面第 0 次循环
这是里面第 1 次循环
这是里面第 2 次循环
```

【代码解析】在以上的程序中，通过两个 for 循环来实现了 3×3 次的循环语句，并且每一个外层循环中都会执行三次内层循环，这可以从打印的结果中看出来。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

注意：在 for 循环中，使用的变量 i 和 j 是定义在 for 循环内部的，其作用域只在本循环内部，也就是只能在所定义的循环内使用，在 for 循环之外就不可以使用了。

【范例 5-7】示例代码 5-7 是一个测试 for 循环的程序。

示例代码 5-7

```
01  public class ForTwoTest {
02
03      public static void main(String[] args) {
04
05          for(int i=1;i<2;i++)
06              System.out.println("这是第一个循环中的 i，i="+i);
07          for(int i=2;i<3;i++)
08              System.out.println("这是第二个循环中的 i，i="+i);
09      }
10
11  }
```

【运行结果】程序运行结果如下。

```
这是第一个循环中的 i，i=1
这是第二个循环中的 i，i=2
```

【代码解析】在以上程序中，分别定义了两个 for 循环（第 5～8 行），并且在两个循环中各有一个 i 变量，但是这两个变量 i 却相互不影响，因为他们的作用域是不同的，第一个 i 的作用域只存在于第一个 for 循环中，第二个 i 的作用域只存在于第二个 for 循环中。

5.3.2 while 循环

while 循环的功能和 for 循环的功能相同，可以将 while 循环看做是 for 循环的简化版本。while 循环的基本格式如下。

```
while (可返回布尔类型值表达式) {
    相关执行语句;
}
```

如果可返回布尔类型值表达式的值是“true”，则会执行相关的语句，然后重新判断该表达式的值，如果值仍为“true”，则重新执行相关的语句，这样一直循环执行到表达式的值为“false”为止。图 5-6 是 while 循环的流程图。

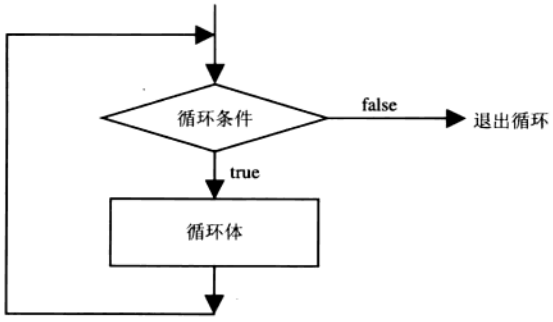


图 5-6　while 循环流程图

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



【范例 5-8】示例代码 5-8 是一个测试 while 语句的程序。

示例代码 5-8

```
01  public class WhileTest {
02      public static void main(String[] args) {
03
04          int i=0;
05          while(i<10){
06              System.out.println(i);
07              i++;
08          }
09
10      }
11  }
```

【运行结果】程序运行结果如下。

```
0
1
2
3
4
...
```

5.3.3　do-while

在 while 循环中，还有另一种表现形式，那就是 do-while，其使用格式如下。

```
do{
    执行语句;
}while(返回为布尔类型值的表达式);
```

在该循环中，首先会执行语句，再判断表达式的值，如果表达式的值为“true”，就会重新去执行语句，然后一直循环直至表达式的值为“false”为止。如图 5-7 是 do-while 循环的流程图。

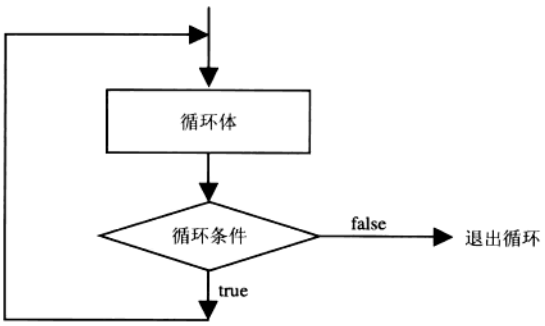


图 5-7　do-while 循环流程图

【范例 5-9】示例代码 5-9 是一个测试 do-while 语句的程序。

示例代码 5-9

```
01  public class DoWhileTest {
02
03      public static void main(String[] args) {
04          int i=0;
05          do{
06              System.out.println(i);
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
07                i++;
08            }while(i<10);
09
10        }
11
12    }
```

程序运行结果如下。

```
0
1
2
3
4
…
```

do-while 和 while 循环的区别是，前者无论表达式的值为何值，都会执行一遍语句，而后者如果表达式的值不为“true”，则不会执行语句。



5.4 中断控制

彩票开奖时，会使用摇号机来随机选择几个号码，在选择完成后，就会中断摇号。在程序中也有中断控制，中断控制语句属于控制流程，其中包括有 continue 和 break，在 continue 语句和 break 语句之后的代码将无法被执行。

5.4.1 continue

continue 语句用于中断程序流程，在用于循环中时，可以中断当前的循环，并进入到下一次的循环中。

【范例 5-10】示例代码 5-10 是一个测试 continue 语句的程序。

示例代码 5-10

```
01  public class ContinueTest {
02
03      public static void main(String[] args) {
04
05          for(int i=0;i<10;i++){
06              if(i==5)
07                  continue;
08              System.out.println("当前执行的是 i="+i);
09
10          }
11
12      }
13
}
```

【运行结果】程序的运行结果如下。

```
当前执行的是 i=0
当前执行的是 i=1
当前执行的是 i=2
当前执行的是 i=3
当前执行的是 i=4
当前执行的是 i=6
当前执行的是 i=7
当前执行的是 i=8
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



```
当前执行的是 i=9
```

【代码解析】以上是执行在 for 循环中打印输出变量 i 的值的一段程序，在其中判断如果 i 的值等于 5，则会中断本次循环，就不会输出 i=5 的结果。

5.4.2　break

break 语句在前面的 switch 循环中已经接触过了，在 switch 语句中，break 语句是用于结束 switch 语句的执行的。在循环中，break 语句用于强制终止循环。

【范例 5-11】示例代码 5-11 是一个测试 break 语句的程序。

示例代码 5-11

```
01  public class BreakTest {
02
03      public static void main(String[] args) {
04
05          for(int i=0;i<10;i++){
06              if(i==6)
07              break;
08              System.out.println("当前执行的是 i="+i);
09
10          }
11      }
12  }
```

【运行结果】程序的运行结果如下。

```
当前执行的是 i=0
当前执行的是 i=1
当前执行的是 i=2
当前执行的是 i=3
当前执行的是 i=4
当前执行的是 i=5
```

【代码解析】在上面的程序中，使用 for 循环来打印输出 i 的值，i 的取值范围是 0～10，在循环中定义了一个判断语句，在 i==6 的时候，强制跳出循环，可以看到程序的运行结果中只打印输出了五个数值。

5.4.3　return

return 语句用于结束当前的方法，并且返回到调用该方法的位置，return 还可以提供相应的返回值。

【范例 5-12】示例代码 5-12 是一个测试 return 语句的程序。

示例代码 5-12

```
01  public class ReturnTest {
02      public boolean var(int i){
03          if(i>0){
04              return true;
05          }
06          return false;
07      }
08
09      public static void main(String[] args) {
10
11          ReturnTest rt=new ReturnTest();
12          if(rt.var(-2)){
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
13                System.out.println("输入的值为正数..");
14            }else{
15                System.out.println("输入的值为负数..");
16            }
17
18        }
19    }
```

【运行结果】程序的运行结果如下。

输入的值为正数..

【代码解析】在程序中实现一个判断一个整数是否为正数的方法（var），在程序中定义了一个返回值类型为布尔类型的方法，方法中包含有两条 return 语句，当 if 条件语句布尔型表达式的结果为 true 时，执行第一条 return 语句，此时退出本方法。创建一个主方法，在主方法中调用 var 方法并传值，并打印输出判断结果。

5.5 习题

【习题 5-1】使用控制语句输出如下的九九乘法表。

1×1=1

2×1=2 2×2=4

3×1=3 3×2=6 3×3=9

4×1=4 4×2=8 4×3=12 4×4=16

5×1=5 5×2=10 5×3=15 5×4=20 5×5=25

6×1=6 6×2=12 6×3=18 6×4=24 6×5=30 6×6=36

7×1=7 7×2=14 7×3=21 7×4=28 7×5=35 7×6=42 7×7=49

8×1=8 8×2=16 8×3=24 8×4=32 8×5=40 8×6=48 8×7=56 8×8=64

9×1=9 9×2=18 9×3=27 9×4=36 9×5=45 9×6=54 9×7=63 9×8=72 9×9=81

【分析】本题主要考查的是读者对循环语句的使用，在此使用双重 for 循环比较简单。首先使用一层 for 循环用于定义循环输出的行，然后在里层再定义一个 for 循环用于定义输出的列。

【主要代码】

```
for (int i = 1;  i <= 9; i++)
   {
     for(int n = 1; n <= i; n++)
     {
       System.out.print( i + "x" + n + "=" + i * n + " ");
     }
     System.out.println();
  }
```

【习题 5-2】编写一个程序，实现课程选择，效果如下。

```
1:数学
2:语文
3:英语
4:物理
5:化学
请输入选择的课程号:
1
您选择的是数学课程
```

【分析】本题主要考查 do-while 循环和 switch 循环的用法，使用 do-while 循环和 switch

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



语句制作一个选课程序，首先要打印出一份选课单列表，然后根据客户的选择输出所选的课程。

【主要代码】

```
do {//do-while 循环
            //打印出选课菜单
            System.out.println("1:数学");
            System.out.println("2:语文");
            System.out.println("3:英语");
            System.out.println("4:物理");
            System.out.println("5:化学");
            System.out.println("请输入选择的课程号：");
            i=(char)System.in.read();          //获得用户输入
            //使用 switch 语句来根据用户输入判断输出结果
switch (i) {
            case '1':
                System.out.println("您选择的是数学课程");
                break;
            case '2':
                System.out.println("您选择的是语文课程");
                break;
            case '3':
                System.out.println("您选择的是英语课程");
                break;
            case '4':
                System.out.println("您选择的是物理课程");
                break;
            case '5':
                System.out.println("您选择的是化学课程");
                break;
            default:
                System.out.println("没有您输入的课程号，请输入 0-5 之间的数字");
                break;
            }
        } while (i<'1'||i>'5');            //输入循环的条件
```

【习题 5-3】使用循环写一个程序，打印出如下的菱形。

```
      #
     ###
    #####
   #######
  #########
 ###########
#############
 ###########
  #########
   #######
    #####
     ###
      #
```

【分析】该题主要考查对循环嵌套的使用，首先要先输出菱形的上半部分，使用双重 for 循环，然后再使用双重 for 循环输出菱形的下半部分。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

【主要代码】

```
if (row >= 1) { // 判断传入的行数
            int x, y;
            for (x = 1; x <= row; x++) {            // 先输出上半部分的正三角形
                for (y = 1; y <= row - x; y++)      // 控制左边输出的空格数
                    System.out.print(" ");
                for (y = 1; y <= 2 * x - 1; y++)    // 控制输出的"#"数
                    System.out.print("#");
                System.out.println();               // 每输出一行就换行一次
            }
            for (x = 1; x <= row; x++) {            // 输出下半部分的正三角形
                for (y = 1; y <= x; y++)   // 控制左边输出的空格数，与上半部分正好相反
                    System.out.print(" ");
                for (y = 1; y <= 2 * (row - x) - 1; y++)    // 控制输出的"#"数
                    System.out.print("#");
                System.out.println(); // 每输出一行就换行一次
            }
        }
```

【习题 5-4】打印 1～100 之间的偶数。

【分析】偶数是能被 2 整除的数。

【主要代码】

```
for (int i = 1; i <= 100; i++) {
   if (i % 2 == 0) {
      System.out.println(i+ "是偶数");
   }
}
```

【习题 5-5】判断 101～200 之间有多少个素数，并输出所有素数。

【分析】素数是只能被 1 和它自身整除的数。

【主要代码】

```
for (int i = 101; i <= 200; i++) {
   for (int j = 2; j <= i; j++) {
      if (i % j == 0 && i == j) {
         sum++;
         System.out.println(i);
      } else if (i % j == 0 && i != j) {
         break;
      }
   }
}
```

【习题 5-6】打印出所有的“水仙花数”，即此数的各个位上数字的三次方之和等于此数。

【分析】利用 for 循环 100～999 之间的数，分解出个位、十位和百位。

【主要代码】

```
      int one, ten, hundred;
      for (int i = 100; i <= 999; i++) {
         one = i % 100 % 10;
         ten = i / 10 % 10;
         hundred = i / 100;
          if (i == (one * one * one + ten * ten * ten + hundred * hundred * hundred)) {
             System.out.println("水仙花数：" +i);
          }
      }
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



【习题 5-7】一个皮球从 100 米高空落下，每次落地后反弹回原高度的一半，再落下。求此球在第十次落地时，共经过多少米，第十次落地后反弹多高？

【分析】对 for 循环的考查。

【主要代码】

```
double h = 100.0, d = h;
for (int i = 1; i <= 10; i++) {
    h = h/2; //第 i 次返回的高度
    d += h;
    if (i == 10) { // 第十次落地时，不计算返回
        d -= h;
    }
}
```

【习题 5-8】现在用一个锅炉烧热水，假设锅炉的容量为 100，当流出的水量超过 100 时，就往外流凉水。

【分析】对 while 循环的考查。

```
public class Boiler {
    public static void main(String[] args) {
        int i = 0;
        while (true) {
            i++;
            if (i > 100) {
                System.out.println("不是开水");
            } else {
                System.out.println("开水");
            }
        }
    }
}
```

【习题 5-9】现在要对第 8 题的 Boiler 进行改造，当热水流完后，不再流凉水，而是进行加热，直到水开再流开水。

【分析】对 while 循环的考查。

```
public class BoilerTwo {
    public static void main(String[] args) {
        boolean isBoil = true;
        int i = 0;
        while (true) {
            if (! isBoil) {
                continue;
            }
            i++;
            if (i < 100) {
                i++;
                System.out.println("开水");
            } else {
                isBoil = false;
            }
        }
    }
}
```

【习题 5-10】写一个类，产生一个 100 以内的随机数，用来模拟红绿灯，当是偶数时，代表绿灯，汽车可以开动，奇数代表红灯，汽车停下。

【分析】对 while 循环的考查。

```
public class TrafficLight {
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
    public static void main(String[] args) throws InterruptedException {
        int x = 200;
        while (x > 0) {
            int i = new Random().nextInt(100);
            if ((i % 2) == 0) {
                i--;
                System.out.println("绿灯，可以继续行驶...");
            } else {
                Thread.sleep(1000);
                System.out.println("红灯，必须停下等待...");
            }
        }
    }
}
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第 6 章　面向对象程序设计

面向对象是 Java 语言的基本设计思想。类是 Java 中最核心最基本的内容。本章主要介绍面向对象编程，首先对面向对象基础进行介绍，使读者对面向对象有一个大体的了解，然后再对类和对象进行深入的介绍。



6.1　面向对象程序设计基础

手机可以看做是一个对象，在手机中存在有成百上千的部件，而每一部件的结构又是比较复杂的，如果手机制造商要在最短的时间内制造出功能完善、性能优越的手机，就需要将手机的各个重要部件分配给不同的部门进行生产，或者承包给其他的生产厂商，如专门生产手机屏幕的厂商、专门生产电池的厂商，等等。面向对象的思想也是如此，程序可以看做一部完整的手机，程序中不同功能的组件就可以看做是不同的部件。

面向对象的思想是基于一整套理论体系构成的，利用封装、继承、多态等方法来实现代码的重用，减少程序错误，进而降低软件开发的工作量。面向对象的语言开发，是一种语言完全或部分地以面向对象的思想进行设计和实现。



6.2　类

无数个细胞组成一个器官，不同的器官组合起来才是一个完整的人，也就是说人体中最基本的元素是细胞。在 Java 语言中，类就充当了细胞的角色。

类是 Java 语言的基础，任何的操作都要在类中完成，如方法、构造函数、参数的传递，等等。实际上，类是 Java 中定义的一个模板。

6.2.1　定义及使用类

类是对象的一般形式，Java 程序的编写，就是将这些对象的共同点抽取出来，然后使用程序语言的形式表达出来。例如，把一个人看做是一个对象，把他作为一个类抽象出来，那么这个人就可以看做是人类的一个对象。类的一般形式如下。

```
class 类名{
类型 变量名;
类型 变量名;
……
类型 方法名(参数){
方法内容
}
……
}
```

在人这个对象中，又有一些属性，如姓名、性别、年龄等。而这个人又可以有吃饭、喝茶这样的动作，所以上面的类就可以写为如下的形式。

```
class People{
String name;
```

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
String sex;
Int age;

void eat(){
System.out.println("我在吃饭");
}
void tea(){
System.out.println("我在喝茶");
}
}
```

有时声明类时，需要在类的前面加上修饰符进行修饰。在 Java 中使用的修饰符主要有以下几种。

- private：只在本类中可见。
- protected：在本类或者同一个包中都可见。
- public：对于任何类都可见。
- 默认（无修饰符）：在本类或者同一个包中的类中都可见。

在类中会定义一些变量，如在类 People 中定义的一些变量如下。

```
String name;
String sex;
Int age;
```

在这些变量中，并没有看到为其加上修饰符进行修饰，这样就相当于使用了默认的修饰符。加上修饰符的变量的示例如下。

```
private String name;
private String sex;
private Int age;
```

方法可以分为有返回类型方法和无返回类型方法两种，如果不需要返回值，可以使用 void 来进行修饰，如果需要有返回值，方法的类型定义必须和返回值的类型相同。例如，要返回一个字符串类型值的方法如下。

```
public String eat(){
    return "我在吃饭";
}
```

使用带有参数的方法一般形式如下。

```
修饰符 类型 方法名(参数类型 参数1,参数类型 参数2,...){
方法体
}
```

在了解完修饰符及方法后，就可以写一个完整的 People 类了，如下所示。

```
public class People {
    private String name;
    private String sex;
    private int age;

    public String eat(String food){
        return "我在吃"+food;
    }

    public void tea(){
        System.out.println("我在喝茶...");
    }

}
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



6.2.2　主方法

一个程序写好之后，如何去运行，从哪里去运行呢？主方法的出现，就解决了这个问题。

在程序中，主方法就是程序的入口。程序若是一间屋子，主方法就是屋子的门。一个由 public 进行修饰的主方法 main 如下，该方法没有返回值，方法中的参数是字符串类型的 args。

```
public static void main(String[] args) {

}
```

在前面的 People 类中，加入主方法，并在主方法中调用 eat 和 tea 方法，就是一个完整的程序了。

【范例 6-1】示例代码 6-1 是一个测试主方法的程序。

示例代码 6-1

```
01  public class Peoplemain {
02      private String name;
03      private String sex;
04      private int age;
05
06      public String eat(String food){
07          return "我在吃"+food;
08      }
09      public void tea(){
10          System.out.println("我在喝茶...");
11      }
12      public static void main(String[] args) {
13          String str=new Peoplemain().eat("苹果");
14          System.out.println(str);
15          new Peoplemain().tea();
16      }
17
}
```

【运行结果】程序运行后，输出结果如下。

```
我在吃苹果
我在喝茶...
```

【代码解析】在本程序中，首先声明了三个属性 name、sex、age，新建一个 eat 方法，并设置 eat 的返回类型为字符串类型，参数为字符串类型。新建一个 tea 方法，在方法中打印输出“我在喝茶..”内容。最后新建一个主方法，在主方法中给 eat 方法传递参数值，并将返回值赋给 str，调用 tea 方法。

6.2.3　构造方法

使用计算器进行新的计算时，为了防止使用到前一次运算时的数据，设计了一个“清空”按钮，进行计算时先按一下该按钮，就可以将前面运算的数据都清除掉，在程序中创建对象时，也需要清除或者修改其属性的值，这就是构造方法。

构造方法是类中比较特殊的一种方法，主要用于初始化一些参数，相当于计算器中的清零。使用构造方法可以把所有的数字变量全都设置为 0，把所有的布尔类型值设置为 false，把所有的对象变量设置为 null。

Java 中提供有默认的构造方法，如果在类中没有提供自己的构造方法，在编译时，编译器就会为这个类添加一个构造方法。构造方法的语法如下。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
People people =new People();
```

在这个语句中，构造方法 People()实际上完成的效果如下。

```
People(){
name=null;
sex=null;
age=0;
}
```

如果在类中设计了自己的构造方法，那么构造方法的名称必须要和类的名称相同，并且没有返回值。

```
public class People {
    private String name;
    private String sex;
    private int age;

    public People(){

    }
}
```

【范例 6-2】示例代码 6-2 是一个测试构造方法的程序。

示例代码 6-2

```
01  public class PeopleConstructor {
02      private String name;
03      private String sex;
04      private int age;
05
06      public PeopleConstructor (){}
07
08      public String eat(String food){
09          return "我在吃"+food;
10      }
11      public void tea(){
12          System.out.println("我在喝茶...");
13      }
14      public static void main(String[] args) {
15
16          PeopleConstructor peoplec=new People();
17          System.out.println("姓名的默认初始值："+peoplec.name);
18          System.out.println("性别的默认初始值："+peoplec.sex);
19          System.out.println("年龄的默认初始值："+peoplec.age);
20      }
21
22  }
```

【运行结果】运行结果如下。

```
姓名的默认初始值：null
性别的默认初始值：null
年龄的默认初始值：0
```

【代码解析】在本程序中，首先声明三个属性 name、sex、age，接着新建一个 eat 方法，并设置 eat 的返回类型为字符串类型，参数为字符串类型。再新建一个 tea 方法，在方法中打印输出“我在喝茶..”内容。最后新建一个主方法，在主方法中调用构造方法，并分别打印出各个属性的初始值（people.name、people.sex、people.age）。

构造函数是用来进行初始化，如果不希望在类中对属性进行初始化，可以在构造函数中加入参数，并通过有参构造方法将值传递给各个属性。有参构造方法的定义如下。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
People(String jname,String jsex;int jage){
name=jname;
sex=jsex;
age=jage;
}
```

在类中提供了自己的构造函数后，在编译时，编译器就不会再提供默认的构造函数了。在创建对象时，就需要添加参数，也就是各属性被初始化的值。创建对象的完整语句示例如下。

```
People people=new People("张三","男",20);
```

【范例 6-3】示例代码 6-3 是一个测试有参构造方法的程序。

示例代码 6-3

```
01  public class PeopleParameter {
02      private String name;
03      private String sex;
04      private int age;
05
06      public PeopleParameter(String jname,String jsex,int jage){
07          name=jname;
08          sex=jsex;
09          age=jage;
10      }
11
12      public String eat(String food){
13          return "我在吃"+food;
14      }
15      public void tea(){
16          System.out.println("我在喝茶...");
17      }
18      public static void main(String[] args) {
19          PeopleParameter people=new PeopleParameter("张三","男",20);
20          System.out.println("姓名："+people.name);
21          System.out.println("性别："+people.sex);
22          System.out.println("年龄："+people.age);
23      }
24  }
```

【运行结果】运行结果如下。

```
姓名：张三
性别：男
年龄：20
```

【代码解析】在本程序中，首先声明三个属性 name、sex、age，新建一个 eat 方法，并设置 eat 的返回类型为字符串类型，参数为字符串类型。新建一个 tea 方法，在方法中打印输出“我在喝茶...”内容。最后新建一个主方法，在主方法中初始化调用有参构造方法并传值，并分别打印出各个属性的初始值。

从以上的程序中可以看到，对象信息已经被初始化，有时候并不需要将所有的信息都初始化，只需要使用其中的一部分属性，这时，就可以使用不同的构造方法进行对象的创建，并增加几个构造函数。

【范例 6-4】示例代码 6-4 是一个测试有参构造方法的程序。

示例代码 6-4

```
01  public class Peoplemore {
02      private String name;
03      private String sex;
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
04      private int age;
05
06      public Peoplemore(String nname,String nsex,int nage){
07          name=nname;
08          age=nage;
09          sex=nsex;
10      }
11
12      public Peoplemore(String jname,int jage){
13          name=jname;
14          age=jage;
15      }
16
17      public static void main(String[] args) {
18          Peoplemore peo=new Peoplemore("张三","男",20);
19          Peoplemore lisi=new Peoplemore("李四",25);
20          System.out.println(peo.name+"的信息：");
21          System.out.println("姓名："+peo.name);
22          System.out.println("性别："+peo.sex);
23          System.out.println("年龄:"+peo.age);
24          System.out.println(lisi.name+"的信息");
25          System.out.println("姓名："+lisi.name);
26          System.out.println("年龄："+lisi.age);
27
28      }
29
30  }
```

【运行结果】运行结果如下。

```
张三的信息：
姓名：张三
性别：男
年龄:20
李四的信息
姓名：李四
年龄：25
```

【代码解析】在本程序中，首先声明三个属性 name、sex、age，创建两个构造方法，在其中设置不同的参数，其中一个对全部的属性进行初始化，另一个只对部分属性值进行初始化，在主方法中初始化调用第一个有参构造方法并传值“张三”的信息，然后调用第二个有参构造方法并传值“李四”的信息，分别打印输出张三和李四的信息。

6.2.4 成员变量

公司一般都会使用到复印机，如果科室多的话，要在每一个科室都配置一台复印机是不大可能的，那么可以在大厅中安装一台公共的复印机，供各个科室共同使用，节省资源也节省成本。在程序中，为了解决这个问题，使用了成员变量，成员变量可以在类中共享，类中的方法都可以使用成员变量。

成员变量又称为全局变量，定义在类中，和类中的方法处于同一个层次。成员变量相当于银行中的取号机，银行中的人都可以对其进行使用。成员变量的语法如下。

```
变量修饰符 类型 变量名;
```

其中，类型可以是任意的数据类型，也可以是数组或类，在一个类中，成员变量是唯一的，但是成员变量的名称可以和类中的方法名相同，如下所示。

```
public class Variable {
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



```
    private int i;
    private int sum;
    public void sum(){
    }
}
```

成员变量可以使用前面介绍到的四种修饰符，作用也是相同的，另外还有一个静态修饰符，也就是使用 static 来修饰变量，在静态方法中使用的变量必须要使用 static 进行修饰。例如，在 main 方法中使用成员变量，就需要写成下面的形式。

```
public class Variable {

    static  int i;
    static  int sum;

    public static void main(String[] args) {
        sum=sum+i;

    }

}
```

如果一个变量前面加上 final 关键字，这个变量就会变成常量，不允许被修改。

【范例 6-5】示例代码 6-5 是一个测试成员变量使用方法的程序。

代码 6-5

```
01  public class Variable {
02      static int i;
03      static private int sum;
04      int j;
05      final int k=10;
06
07      public void sum(){
08          sum=0;
09          for(j=1;j<11;j++){
10
11              sum=sum+j;
12          }
13          System.out.println("从 1 到 10 累计相加结果："+sum);
14          System.out.println("常量 k 的值是"+k);
15      }
16
17      public static void main(String[] args) {
18          sum=0;
19          for(i=1;i<11;i++){
20
21              sum=sum+i;
22          }
23          System.out.println("从 1 到 10 累计相加结果："+sum);
24
25          Variable var=new Variable();
26          var.sum();
27
28      }
29
30  }
```

【运行结果】运行结果如下。

```
从 1 到 10 累计相加结果：55
从 1 到 10 累计相加结果：55
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

常量 k 的值是 10

【代码解析】在本程序中，首先声明成员变量 i、sum、j、k，并分别使用不同的修饰符进行修饰，创建和变量 sum 名相同的方法，并在方法中使用所声明的变量来进行循环累加，并打印输出累加的结果。创建主方法，并在主方法中使用 static 修饰的变量来完成循环累加，然后打印输出，最后调用 sum 方法。

6.2.5 局部变量

现在各个公司都有电话，如果公司比较大、部门比较多的话，设置一个公共的电话会降低效率，而且这样的话，恐怕电话会一直响个不停，要解决这个问题，可以至少为每一个部门配置一部电话。在程序中，为了解决类似的问题，就出现了局部变量，局部变量只能在方法中使用。

局部变量是在方法中定义和使用的，它的使用范围仅限于所声明的方法内。在不同的方法内所定义的变量名可以相同，变量的类型可以不同。这也就相当于在两个不同的城市中各有一个名为张三的人，二者是相对独立的，而且这两个张三的职业也是不同的，如一个是公务员，另一个是个体经营者。

【范例 6-6】示例代码 6-6 是一个测试局部变量使用方法的程序。

代码 6-6

```
01  public class VariablePart {
02      public void eat() {
03          int i = 12;
04          String eat = "吃饭时间";
05          String tea = "下午茶时间";
06          if (i > 11 && i < 14) {
07              System.out.println("现在时间是" + i + "点，是" + eat);
08          } else if (i > 15 && i < 17) {
09              System.out.println("现在时间是" + i + "点，是" + tea);
10          }
11
12      }
13
14      public void tea() {
15          int i = 16;
16          String eat = "吃饭时间";
17          String tea = "下午茶时间";
18          if (i > 11 && i < 14) {
19              System.out.println("现在时间是" + i + "点，是" + eat);
20          } else if (i > 15 && i < 17) {
21              System.out.println("现在时间是" + i + "点，是" + tea);
22          }
23      }
24
25      public static void main(String[] args) {
26          VariablePart vp = new VariablePart();
27          vp.eat();
28          vp.tea();
29      }
30  }
```

【运行结果】运行结果如下。

```
现在时间是 12 点，是吃饭时间
现在时间是 16 点，是下午茶时间
```

【代码解析】在程序中，创建了两个方法，分别是 eat 和 tea，并在方法中声明了同样的三

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



个变量 i、eat、tea，新建两个方法 tea 和 eat，在其中分别为三个变量赋初始值，然后使用 if 语句进行判断，如果时间在 11 点和 14 点之间，则输出字符串“吃饭时间”，如果时间在 15 点和 17 点之间，则输出“下午茶时间”。最后创建主方法，并在主方法中调用 eat 和 tea 方法。

6.2.6　this 关键字

this 是 Java 中的一个关键字，使用 this 关键字可以不通过对象引用成员变量和方法，这正如去某个公司找领导，不需要先找他的秘书，直接拨打该领导的手机就可以。下面是一个使用 this 关键字和使用对象引用成员变量的示例。

【范例 6-7】示例代码 6-7 是一个测试 this 关键字使用方法的程序。

代码 6-7

```
01  public class ThisTest {
02      private String eat="正在吃饭...";
03      private String tea="正在喝茶...";
04      private String work="正在工作....";
05
06      public void zhagnsna(){
07          System.out.println("使用 this 关键字来引用成员变量，各变量输出为：");
08          System.out.println("张三"+this.eat);
09          System.out.println("张三"+this.tea);
10          System.out.println("张三"+this.work);
11
12      }
13
14      public void lisi(){
15          System.out.println("不使用 this 关键字来引用成员变量，各变量输出为：");
16          System.out.println("李四"+eat);
17          System.out.println("李四"+tea);
18          System.out.println("李四"+work);
19
20      }
21      public static void main(String[] args) {
22          new ThisTest().zhagnsna();
23          new ThisTest().lisi();
24
25      }
26
27  }
```

【程序结果】程序的运行结果如下。

```
使用 this 关键字来引用成员变量，各变量输出为：
张三正在吃饭...
张三正在喝茶...
张三正在工作....
不使用 this 关键字来引用成员变量，各变量输出为：
李四正在吃饭...
李四正在喝茶...
李四正在工作....
```

【程序解析】在程序中，首先创建了三个成员变量 eat、tea、work，并分别赋初值，创建两个方法 zhangsna 和 lisi，在这两个方法中，前者使用了 this 关键字来打印输出张三的动作，后者未使用 this 关键字来打印输出李四的动作。最后创建一个主方法，并在主方法中分别调用 zhagnsna 和 lisi 方法。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

从上面的例子中可以看出，一般情况下，this 关键字可有可无，但是如果成员变量和方法中的参数名相同的时候，this 关键字就显得至关重要了。例如，去某公司找张三，前台告诉你公司有两个张三，要找的是哪一个呢？突然想起来，这里有张三的手机号，还是直接给他打电话，让他自己出来吧。

【范例 6-8】示例代码 6-8 是一个测试 this 关键字使用方法的程序。

代码 6-8

```
01  public class ThisTesttwo {
02      String eat="吃饭时间";
03      String tea="下午茶时间";
04      int i=16;
05      public void eat(int i){
06
07          if(i>11&&i<14){
08              System.out.println("现在时间是"+i+"点，是"+eat);
09          }else if(i>15&&i<17){
10              System.out.println("现在时间是"+i+"点，是"+tea);;
11          }
12
13      }
14      public void tea(int i){
15
16          if(this.i>11&&this.i<14){
17              System.out.println("现在时间是"+this.i+"点，是"+eat);
18          }else if(this.i>15&&this.i<17){
19              System.out.println("现在时间是"+this.i+"点，是"+tea);;
20          }
21      }
22
23      public static void main(String[] args) {
24          ThisTesttwo vp=new ThisTesttwo();
25          vp.eat(12);
26          vp.tea(12);
27      }
28
29  }
```

【运行结果】程序的运行结果如下。

```
现在时间是 12 点，是吃饭时间
现在时间是 16 点，是下午茶时间
```

【代码解析】在程序中，首先声明了三个成员变量 eat、tea 和 i，接着新建两个方法 eat 和 tea，其中各有一个整型参数 i，和成员变量名称相同。在方法中，判断时间，在 eat 方法中使用成员变量 i，也就是使用 this 关键字，在 tea 方法中使用参数 i，并分别输出“吃饭时间”和“下午茶时间”。最后创建主方法，在主方法中调用 eat 和 tea 方法，并分别传递参数为 12。

从以上的程序中可以看出，在两个方法中都有参数，并且两个方法中所传递的值也是相同的，但第一个方法中有 this 关键字，也就是使用了成员变量，在该方法中 i 的值是 16，第二个方法中没有使用 this 关键字，也就是使用了方法中的参数，在该方法中 i 的值是 12。

设计构造函数时，有时会设计多个构造函数来满足不同条件下的初始化，有时要在一个构造函数中调用另一个构造函数，而构造函数之间的差别也仅在于其中的参数，使用 this 关键字时，编译器可以根据其中的参数来寻找不同的构造方法。

【范例 6-9】示例代码 6-9 是一个测试 this 关键字使用方法的程序。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



代码 6-9

```
01  public class ThisConstructor {
02      String name;
03
04      public ThisConstructor(String onename){
05          name=onename;
06          System.out.println("姓名："+name);
07      }
08
09      public ThisConstructor(){
10          this("张三");
11      }
12
13      public static void main(String[] args) {
14
15          new ThisConstructor();
16      }
17  }
```

【运行结果】程序的运行结果如下。

```
姓名：张三
```

【代码解析】在程序中，首先声明一个成员变量 name，创建两个构造方法，第一个构造方法中有字符串类型的参数 onename，第二个构造方法中无任何参数。在第一个构造方法中使用 this 关键字为 name 赋值，然后打印输出姓名；在第二个构造方法中调用第一个构造方法，并为第一个构造方法传值为“张三”。最后创建一个主方法，并在主方法中创建 ThisConstructor 对象并初始化。



注意：不能够使用相同的方法来调用两个不同的构造方法，并且被调用的构造方法要放在方法的最开始位置。



6.3　对象

日常生活中，对象可以是任何的物体，如杯子、桌子、电冰箱等。一个抽象的事物也可以看做是一个对象，如一个想法，它虽然看不见也摸不到，但是可以通过人类的语言体现出来。

6.3.1　创建和使用对象

在朋友那里看到一个比较不错的杯子，自己也想拥有一个，可是始终没有在商店里找到一模一样的，还好现在什么东西都可以进行定制，那么就让制造商给量身定做一个吧！在这个示例中，制作这个杯子也就是对象的创建。

在类中，创建一个对象要使用 new，也就是新建。创建对象的语法一般如下。

```
类名 对象名= new 类名(参数列);
```

【范例 6-10】示例代码 6-10 是一个创建对象的程序。

代码 6-10

```
01  public class CreateObjectTest {
02      public static void main(String[] args) {
03          Cup cup=new Cup();
04          System.out.println("对杯子没有要求");
05          Cup cupch=new Cup("红色",15);
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
            System.out.println("一个高为 15 厘米的红色杯子");
            Cup cupph=new Cup(10.0,15);
            System.out.println("一个高为 15 厘米，价格为 10 元的杯子");
            Cup cuppc=new Cup(10.0,"红色");
            System.out.println("一个 10 元的红色杯子");
            Cup cuppch=new Cup(10.0,"红色",15);
            System.out.println("一个价格为 10 元高为 15 厘米的红色杯子");
        }

}
class Cup{
    private int high;
    private String color;
    private double price;

    public Cup(){

    }
    public Cup(String color,int high){
        this.color=color;
        this.high=high;
    }
    public Cup(double price,int high){
        this.price=price;
        this.high=high;

    }
    public Cup(double price,String color){
        this.price=price;
        this.color=color;
    }
    public Cup(double price,String color,int high){
        this.price=price;
        this.color=color;
        this.high=high;
    }

}
```

【运行结果】程序的运行结果如下。

```
对杯子没有要求
一个高为 15 厘米的红色杯子
一个高为 15 厘米，价格为 10 元的杯子
一个 10 元的红色杯子
一个价格为 10 元高为 15 厘米的红色杯子
```

【代码解析】在程序中，创建一个 Cup 类代表杯子，并声明三个成员变量 high（高）、color（颜色）、price（价格），在 Cup 类中创建五个构造方法，第一个构造方法中没有参数，第二个构造方法中有 color 和 high 两个参数，在第三个构造方法中有 price 和 high 两个参数，在第四个构造方法中有 price 和 color 两个参数，在第五个构造方法中有三个参数 price、high 和 color，在类 CreateObjectTest 中创建主方法，并在主方法中利用五个不同的构造方法来创建五个对象，然后打印输出。

在得到给出杯子的模型和一些参数后，觉得还不大满意，既然是定制，那就按照自己的要求来，重新设置杯子的一些参数。在类中，使用 new 关键字来得到类和新对象的关联，然后就可以对新对象进行操作了。例如，在例子中应用 Cup 来获得一个类 Cup 的对象实体，然后就可以对 Cup 中的属性值进行赋值操作了（需要将属性设置为 public）。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



【范例 6-11】示例代码 6-11 是一个创建对象的程序。

代码 6-11

```
01  public class CupTest {
02      public String color="绿色";
03      public int high=12;
04      public double price=10.0;
05
06      public CupTest(){
07
08      }
09      public CupTest(String color,int high,double price){
10          this.color=color;
11          this.high=high;
12          this.price=price;
13      }
14      public void water(){
15          System.out.println("使用杯子喝水..");
16      }
17
18  }
19
20  public class TestCup {
21      public static void main(String[] args) {
22          CupTest ct=new CupTest();
23          ct.high=15;
24          ct.color="红色";
25          ct.price=10.0;
26          System.out.println("一个价格为"+ct.price+
27                  "元、高为"+ct.high+"厘米的"
28                  +ct.color+"杯子");
29          ct.water();
30      }
31
32  }
```

【运行结果】程序的运行结果如下。

```
一个价格为 10.0 元、高为 15 厘米的红色杯子
使用杯子喝水..
```

【代码解析】在程序中，创建了两个类 CupTest 和 TestCup，在第一个类中，声明三个成员变量 color、high 和 price，并分别为赋值，然后再创建两个构造方法，第一个构造方法中没有任何参数，第二个参数中有全部的三个参数，创建一个 water 方法，用于打印输出“使用杯子喝水..”字样。在第二个类中，新建一个主方法，并在主方法中创建类 CupTest 的对象 ct，然后分别为三个参数重新赋值，最后将信息打印输出。

6.3.2　对象行为和属性的访问

每一个人都会有身高、年龄、家庭住址等信息，这些信息就是人这个对象的属性，每一个人都会做一些事情，如吃饭、喝水、打字等，这些就是行为，如果想了解一个人的属性信息，就问“您贵姓？”、“您住哪？”，如果想要知道这个人的行为，可以问“您在干嘛？”。

在程序中，如何得知对象的属性和行为呢？在获得一个类的对象实体后，在对象实体后加“.”，就可以调用对象的属性，如果想得到对象的行为，也可以采用同样的方式。

【范例 6-12】示例代码 6-12 是一个访问对象行为和属性的程序。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

代码 6-12

```
01  public class People {
02      public String name="张三";
03      public String sex="男";
04      public int age=23;
05      public String address="北京市海淀区";
06
07      public People(){
08
09      }
10      public People(String name,String sex,int age,String address){
11          this.name=name;
12          this.sex=sex;
13          this.age=age;
14          this.address=address;
15      }
16      public String  eat(){
17          return "正在吃饭...";
18
19      }
20
21  }
22
23  public class PeopleTest {
24      public static void main(String[] args) {
25          People people=new People();
26          System.out.println("个人信息输出：");
27          System.out.println("姓名："+people.name);
28          System.out.println("性别："+people.sex);
29          System.out.println("年龄："+people.age);
30          System.out.println("住址："+people.address);
31          System.out.println(people.name+"在干嘛？");
32          System.out.println(people.name+people.eat());
33          }
34
}
```

【运行结果】程序运行结果如下。

```
个人信息输出：
姓名：张三
性别：男
年龄：23
住址：北京市海淀区
张三在干嘛？
张三正在吃饭...
```

【代码解析】在程序中创建了两个类 People 和 PeopleTest。在 People 类中，声明了四个属性 name、sex、age 和 address，并分别赋值，然后创建了两个构造方法，并创建了一个有返回类型的 eat 方法，并返回字符串“正在吃饭...”。在 PeopleTest 类中，创建一个主方法，并创建 People 类的对象 people，然后分别获得 people 的属性，并打印输出，最后打印输出 People 类中的 eat 方法返回值。

6.3.3 比较对象

讨论某一个人的好坏，前提是得有一个参照的对象，把这个人和参照的标准进行比较，然后才能得出结论，这也就是对象的比较。在程序中，使用“==”和 equals 方法来进行对象的比较。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



【范例 6-13】示例代码 6-13 是一个比较对象的程序。

代码 6-13

```
01  public class Equal {
02      public static void main(String[] args) {
03          String stra=new String("你好");
04          String strb=new String("你好");
05          String strc=stra;
06
07          boolean a=stra==strb;
08          boolean b=stra==strc;
09
10          boolean c=stra.equals(strb);
11          boolean d=stra.equals(strc);
12
13          System.out.println("使用“==”进行比较结果：");
14          System.out.println("stra和strb进行比较:"+a+"\n"+"stra和strc进行比较:
"+b+"\n");
15          System.out.println("使用“equal”方法进行比较结果：");
16          System.out.println("stra和strb进行比较:"+c+"\n"+"stra和strc进行比较:
"+d+"\n");
17      }
18}
```

【程序结果】程序的运行结果如下。

```
使用“==”进行比较结果：
stra 和 strb 进行比较：false
stra 和 strc 进行比较：true

使用“equal”方法进行比较结果：
stra 和 strb 进行比较：true
stra 和 strc 进行比较：true
```

【程序解析】在程序中，首先创建主方法，在主方法中创建字符串类型变量 stra、strb 和 strc，其中，为 stra 和 strb 分别赋值为“你好”，strc 则赋值为 stra。创建布尔类型变量 a、b、c、d，并分别将 stra、strb、strc 使用“==”方法和 equal 方法所比较的结果赋值给 a、b、c、d。最后打印输出比较的结果。

从以上的程序结果中可以看出，使用“==”判断的是二者是否为同一个对象，比较的是两个对象的地址，即便两个对象的值是相同的，地址不同也会输出 false 的。使用 equal 比较的是两者的值，即便不是同一个对象，只要值相同，就会输出 true。

6.3.4　销毁对象

我们在日常生活中难免会产生一些垃圾，这些垃圾一般都会被放置到固定的垃圾箱中，但是如果垃圾箱被装满了呢？那就会将它们送到某一个地方进行掩埋或者是烧掉，这样垃圾箱就又有空间了。在 Java 中，这就是垃圾回收机制。

垃圾回收机制（GC）在 Java 中的使用方法如下。

```
System.gc();
```

垃圾回收机制的使用非常简单，只需要调用上面的方法就可以了。垃圾回收机制可以使程序员不必去考虑垃圾回收的问题，可以大大地降低程序员的代码量，垃圾回收机制用来管理使用 new 关键字创建的对象所占用的内存资源。垃圾回收机制可以管理内存的资源，使内存资源不会发生泄漏。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

6.4 习题

【习题 6-1】输出两个人的信息，在获取这两个人的信息时，使用了类中的成员变量，一个是 ClassDemo_01 girl，另一个是 ClassDemo_01 boy，再通过 Friends 方法对这两个变量进行赋值。

【分析】该题主要考查的是成员变量的使用。

【核心代码】

```
class Friends {
    // 定义类变量
    ClassDemo_01 girl;
    ClassDemo_01 boy;
    public Friends() {// 利用构造方法为类变量赋值
        String gname = "Anni";
        System.out.println("My Girlfriend's name is : " + gname);
        String gphoneNum = "13545681233";
        System.out.println("My Girlfriend's name is : " + gphoneNum);
        String gemail = "Anni129@sina.com";
        System.out.println("My Girlfriend's name is : " + gemail);
        String gaddress = "英国伦墩";
        System.out.println("My Girlfriend's name is : " + gaddress);
        System.out.println("创建并实例化 ClassDemo_01 对象 girl");
        girl = new ClassDemo_01(gname, gphoneNum, gemail, gaddress);// 创建并实例
                                                                    //化类变量
        girl.printMessage();// 调用该类变量的方法
        System.out.println();
        String bname = "Tom";
        System.out.println("My Boyfriend's name is : " + bname);
        String bphoneNum = "13255638897";
        System.out.println("My Boyfriend's name is : " + bphoneNum);
        String bemail = "Tomgood@vip.com";
        System.out.println("My Boyfriend's name is : " + bemail);
        String baddress = "美国洛杉矶";
        System.out.println("My Boyfriend's name is : " + baddress);
        System.out.println("创建并实例化 ClassDemo_01 对象 boy");// 创建并实例化类变量
        boy = new ClassDemo_01(bname, bphoneNum, bemail, baddress);// 调用该类
                                                                   //变量的方法
        boy.printMessage();
    }
}
```

【习题 6-2】以一个通讯录为例，介绍类的基本组成结构和使用方法。在该通讯录中，可以存储姓名、联系电话、E-mail 和家庭住址。

【分析】本题主要考查的是类的组成，以及类中方法和变量的使用。

【核心代码】

```
String name;              // 姓名
    String phoneNum;      // 联系电话
    String email;         // E-mail
    String address;       // 家庭住址
    // 利用该类的构造方法为其成员变量赋值
    public ClassDemo_01(String name, String phoneNum, String email,
            String address) {
        this.name = name;
        this.phoneNum = phoneNum;
        this.email = email;
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
        this.address = address;
    }
    public void printMessage() {// 定义该类的方法
        System.out.println("姓名:    " + name);
        System.out.println("联系电话:    " + phoneNum);
        System.out.println("电子邮箱地址: " + email);
        System.out.println("目前居住地址: " + address);
    }
    public static void main(String[] args) {
        // 用关键字 new 创建该类的实例化对象，为成员变量赋值
        ClassDemo_01 cd = new ClassDemo_01("张小红", "13841146948",
                "zhangxiaohong@sina.com", "北京昌平区沙河镇 251 号");
        // 调用方法
        cd.printMessage();
    }
```

【习题 6-3】编写一个学生类，包括姓名、性别、年龄属性，并在构造函数中初始化这三个属性，注意保留无参数构造函数。

【分析】本题主要考查的是类的带参数构造函数及 this 变量的使用。

【核心代码】

```
public class Student {
   private String name;
   private String gender;
   private int age;

   public Student() {
   }
   public Student(String name, String gender, int age) {
      this.name = name;
      this.gender = gender;
      this.age = age;
   }
}
```

【习题 6-4】为第 3 题的 Student 类添加读和写的行为，这两个行为要将输入内容转化后输出。

【分析】本题主要考查的是类的方法和局部变量的使用。

【核心代码】

```
public String read(String content) {
   String mine = "I learned early.";
   String remain = mine + content;
   System.out.println("这是我学到的内容: " + remain);
   return remain;
}
public String write(String content) {
   System.out.println("这书写的内容: " + content);
   return content;
}
```

【习题 6-5】创建一个 Teacher 类，检查第 3 题中 Student 对象的学习情况。

【分析】本题主要考查的是对象的创建与使用。

【核心代码】

```
public class Teacher {
   String content = "This is the knowlege to teach.";
   public void check(Student student) {
      student.read(content);
      student.write(content);
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
    }
}
```

【习题 6-6】现有两名学生，两名学生向教师介绍自己后，教师对这两名学生进行比较。

【分析】本题主要考查的是对象的使用与比较，为 Student 类增加介绍的功能，为 Teacher 类增加比较的功能。

【核心代码】

```
public String introduce() {
    return "My name is " + name + ". I am a " + gender + ", and my age is " + age
+ ".";
}

public void manage() {
    Student a = new Student("Andi", "girl", 20);
    Student b = new Student("Jack", "boy", 22);
    a.introduce();
    b.introduce();
    System.out.println("学生 a 和学生 b 是同一个人吗? "+ (a == b));
}
```

【习题 6-7】构造一个学校类，学校里有学生和教师进行教学活动。

【分析】本题主要考查的是主函数和对象的创建与使用，为 Teacher 类增加一个 teach()行为，为 Student 类增加一个 study()行为。

【核心代码】

```
public String teach() {
    return "content";
}

public void study(String content) {
    System.out.println("我正在学习："+ content);
}

public static void main(String[] args) {
    Teacher teacher = new Teacher();
    Student studentA = new Student();
    Student studentB = new Student();
    String content = teacher.teach();
    studentA.study(content);
    studentB.study(content);
}
```

【习题 6-8】构造一个类 Dog，描述狗这种动物。

【分析】本题主要考查的是类的设计。

【核心代码】

```
public class Dog {
    private String name;
    private String kind;
    private String weight;
    public Dog(String name) {
        this.name = name;
    }
    public Dog(String name, String kind, String weight) {
        this.name = name;
        this.kind = kind;
        this.weight = weight;
    }
    public void eat() {
        System.out.println("Wating...");
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



```
    }
    public void loud() {
        System.out.println("Barking...");
    }
    public void run() {
        System.out.println("Moving...");
    }
    public void tease() {
        System.out.println("Amusing...");
    }
}
```

【习题 6-9】构造一个类，在其主方法中打印成员变量和调用方法。

【分析】本题主要考查对象的创建和成员变量以及方法的调用。

【核心代码】

```
public class Car {
    public String color;
    public String number;
    public String brand;

    public Car(String color, String number, String brand) {
        this.color = color;
        this.number = number;
        this.brand = brand;
    }

    public void run() {
        System.out.println("Running...");
    }

    public static void main(String[] args) {
        Car car = new Car("black", "6688", "BMW");
        car.run();
        System.out.println("汽车颜色: "+ car.color);
        System.out.println("汽车牌号: "+ car.number);
        System.out.println("汽车品牌: "+ car.brand);
    }
}
```

【习题 6-10】实现第 9 题 Car 类的 equals 方法，对 Car 对象进行比较。

【分析】本题主要考查对象的比较。

【核心代码】

```
        System.out.println("car 与 c1 是否是同一辆车: " + car.equals(c2));
        System.out.println("c1 与 c2 是否是同一辆车: " + c1.equals(c2));
        System.out.println("c1 与 c3 是否是同一辆车: " + c1.equals(c3));
    public boolean equals(Object o) {
        if (null == o) return false;
        Car c = (Car) o;
        return this.color.equalsIgnoreCase(c.color) && this.number.equalsIgnoreCase
(c.number) && this.brand.equalsIgnoreCase(c.brand);
    }
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第 7 章　继　　承

面向对象的程序设计扩展了基于对象的程序设计，可以提供类型/子类型的关系，子类不再需要重新实现所有的特征，而是继承其父类的数据成员和成员变量。本章主要介绍继承及其继承关系、类的关系、抽象类和派生类等内容。



7.1　继承概述

“子承父业”可以用来说明继承，即孩子可以继承父亲的一些特征，以及父亲的事业、资产等。本节主要概述继承。

7.1.1　什么是继承

在程序中的继承是指若一个类从另一个类派生而来，派生类自动具有了被继承类所有的特性。其中，被继承的类称为“父类”，继承父类的类就叫做“子类”，也就是父亲和孩子。有时也称父类为超类或基类，称子类为扩展类或者是派生类。

使用继承可以使得子类具有父类的各种属性和方法，也就是父亲的“事业资产”等，从而不需要再次编写相同的代码，而且，在程序中可以大大地促进软件的复用，并且集成的形式是可以更加复杂的。

子类在继承父类后，也可以拥有自己的一些特性，可以使得该子类更加具体、功能强大。例如，汽车、小汽车、货车，小汽车和货车可以看做是汽车的子类，在小汽车和货车中继承了所有汽车的功能，如刹车、加速等功能，但是在小汽车和货车中还有各自不同的特点。

继承可分为单一继承和多重继承，单一继承类似于一个“独生子”，而多重继承则可以理解为一个父亲有多个孩子。其中，单一继承是最常见的继承方式，实现也比较简单。多重继承虽然可以提供更加灵活强大的功能，但是其语法比较复杂，实现也比较困难。

7.1.2　实现继承

实现继承也就是创建一个类的子类，需要在类的声明中加入 extends 子句，其格式如下。

```
<修饰符> class <子类名> extends <超类名>
{
    //实现代码
}
```

其中，参数 class 是关键字。子类名可以由用户随意定义，但必须要符合 Java 标识符中的命名规范，超类名也就是被继承的父类名，示例如下。

```
public class Son extends Father{

}
```



注意：在 Java 中，一个类只有一个父类，也就相当于是一个孩子只有一个亲生父亲。一个 extends 只能够指定一个类。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



【范例 7-1】示例代码 7-1 是一个测试继承的程序。

示例代码 7-1

```
01  public class MiniCar extends Car{
02      public static void main(String[] args) {
03          MiniCar car=new MiniCar();
04          System.out.println("父类中车的颜色是："+car.color);
05          car.setColor("blue");
06          System.out.println("子类中车的颜色是："+car.color);
07      }
08  }
09  class Car {
10      String color="red";
11      public void setColor(String c){
12          this.color=c;
13      }
14  }
```

【运行结果】程序运行结果如下。

```
父类中车的颜色是：red
子类中车的颜色是：blue
```

【代码解析】在以上的程序中，实现的是一个比较简单的继承。首先定义了 Car 类，其中包含有字符串类型的 color，public 类型的方法 setColor。然后定义了类 MiniCar，代表小汽车。类 MiniCar 继承自类 Car，作为 Car 的子类，类 MiniCar 没有定义自己的任何属性，方法也仅仅只有一个 main 方法。在 main 方法中，创建了类 Car 的一个对象 car，首先打印 car 的属性 color，打印结果为 red，然后，调用继承自类 Car 的属性 setColor，最后再次打印 car 的属性 color，打印结果为 blue。



注意：在 Java 中，子类无法继承父类中使用 private 修饰的类成员。如果将 color 的属性定义为 private，则语句会报错。



7.2　继承关系

7.2.1　单一继承关系

单一继承关系，一个类只继承一个直接父类，这种也是比较简单的一种继承关系。

【范例 7-2】示例代码 7-2 是一个测试单一继承的程序。

示例代码 7-2

```
01  public class MiniCar2 extends Car {
02      private int maxspeed;
03
04      public static void main(String[] args) {
05          MiniCar2 car = new MiniCar2();
06          System.out.println("父类中车的颜色是：" + car.color);
07          car.setColor("blue");
08          System.out.println("子类中车的颜色是：" + car.color);
09
10          System.out.print("小汽车的最大速度是：");
11          car.setMaxspeed(180);
12          System.out.println(car.maxspeed);
13
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
    }

    public int getMaxspeed() {
        return maxspeed;
    }

    public void setMaxspeed(int maxspeed) {
        this.maxspeed = maxspeed;
    }
}
```

【运行结果】运行结果如下。

```
父类中车的颜色是：red
子类中车的颜色是：blue
小汽车的最大速度是：180
```

【代码解析】在以上程序的中，实现了单一继承关系，并在继承的同时，实现了类的扩展，该示例和示例 7-1 类似，同样先定义一个 Car 类，即代表了各种汽车的基类。类 Car 中包含了 String 类型的属性 color，public 类型的方法 setColor。然后定义了一个 MiniCar 类，代表小汽车。类 MiniCar 同样继承了父类 Car 的属性 Color 和 setColor 方法，除了继承父类外，类 MiniCar 还定义了自己的属性 maxspeed 以及相应的获取与设置的方法 getMaxspeed、setMaxspeed。main() 方法中先创建了类 MiniCar 的一个对象 car，然后调用父类的属性和方法，并且进行打印，然后调用 MiniCar 自身的属性方法进行打印。

7.2.2 多重继承关系

在 Java 中，一个类只可以继承一个父类，即只有单一继承，没有多重继承。但可以通过实现接口的方法来达到多重继承的目的。

【范例 7-3】示例代码 7-3 是一个测试多重继承的程序。

示例代码 7-3

```
public class SonTest extends FatherTest implements Stepfather {
    public void hitChild() {
        System.out.println("这是继父");
    }

    public static void main(String[] args) {
        SonTest son = new SonTest();
        son.pet("这是生父");
        son.hitChild();
    }
}

class FatherTest {
    public void pet(String str) {
        System.out.println(str);
    }
}

interface Stepfather {
    public void hitChild();
}
```

【运行结果】程序运行结果如下。

```
这是生父
这是继父
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

【代码解析】在以上的程序中，首先定义了一个 FatherTest 类，作为父类，并在类中实现一个方法 pet，实现打印输出。然后定义一个接口 Stepfather，并在其中定义一个方法 hitChild()。最后创建类 SonTest，继承类 FatherTest 并实现接口 Stepfather，在该类中实现接口中的 hitChild 方法，最后打印输出。

7.3　抽象类

抽象类是 Java 中比较特殊的类，其中只包含有对方法的声明而没有具体的实现。在抽象类中，对所有的方法都只进行了声明而没有具体的实现。

抽象类中的方法需要声明方法的名称，但不用指明方法体。当要对问题域进行分析和设计，得出抽象概念时可以使用抽象类。抽象类要比类的继承更加高级一些，是对类的抽象。抽象类拥有未能够实现的方法，而且不能够被实例化，只能被继承扩展。在 Java 中抽象类的语法如下。

```
abstact class ClassName{
        abstract ReturnType abstractMethodName(paramList);
        //其他方法代码
}
```

在以上的语法代码中，abstract 是关键字，表示此类是抽象类。class 也是关键字。Classname 则是抽象类的类名。类中的“abstract”表示该方法是一个抽象方法。ReturnType 表示抽象方法的返回类型，为合法的 Java 数据类型。abstractMethodName 表示抽象方法的方法名。

【范例 7-4】示例代码 7-4 是一个测试定义抽象类的程序。

示例代码 7-4

```
01  public class SubClass extends AbstractClass {
02     public void speak() {
03        System.out.println("我请大家去吃饭...");
04     }
05
06     public static void main(String[] args) {
07        SubClass sc = new SubClass();
08        sc.speak();
09        sc.eat();
10     }
11  }
12
13  abstract class AbstractClass {
14     public void eat() {
15        System.out.println("我们一起去吃饭...");
16     }
17
18     public abstract void speak();
19  }
```

【运行结果】程序运行结果如下。

```
我请大家去吃饭...
我们一起去吃饭...
```

【代码解析】在程序中，首先定义了一个抽象类，类名为 AbstractClass。在该类中包含有两个方法，一个为一般的方法 eat()，另一个则是抽象方法 speak()。需要注意的是，在 eat()方法中包含有方法体，有具体的方法实现过程，而 speak()则被 abstract 修饰，并没有方法体，只

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

是定义了方法的结构。然后定义了类 SunClass，该类继承了抽象类 AbstractClass。并且在该类中实现了抽象类中的抽象方法 speak()，最后在 main()方法中创建了类 SubClass 的对象，并分别调用 eat()和 speak()方法。

通过对以上的示例和语法的介绍，可以总结抽象类的特点如下。

- 至少含有一个抽象方法。
- 不能被实例化。
- 继承抽象类的子类必须实现其所有抽象方法才能够被实例化，否则该子类还是抽象类。



7.4 使用关键字 final

final 是 Java 中一个比较重要的关键字，可以用它来修饰成员变量，也可以用它来修饰方法，甚至还可以修饰一个类。如果成员变量被 final 修饰的话，这个成员变量就变成一个常量；如果一个方法被 final 修饰，则这个方法不能被重写；如果一个类被 final 修饰，则这个类不能被继承。

7.4.1 常量

在一个变量前使用 final 进行修饰，这个变量就会变成一个常量，也就是该常量只能够被赋值一次，可以不在它声明的那一行赋值，但是需要在构造函数完成之前保证这个变量有一个值。

【范例 7-5】示例代码 7-5 是一个测试 final 常量的程序。

示例代码 7-5

```
01  public class Constant {
02     public static void main(String[] args) {
03        ConstantTest ct = new ConstantTest();
04        System.out.println("a 的值是: " + ct.print());
05
06     }
07  }
08
09  class ConstantTest {
10     int print() {
11        final int a = 10;
12        return a;
13     }
14  }
```

【运行结果】程序运行结果如下。

```
a 的值是: 10
```

【代码解析】在程序中，首先创建一个类 ConstantTest 类，并在其中创建方法 print()，设置一个变量 a，使用 final 修饰。然后创建类 Constant，在其中创建 ConstantTest 的对象 ct，并调用 print 方法，打印输出 a 的值。

如果定义为 final 的变量是静态变量，那它的赋值最晚不得迟于最后一个静态语句块的完成。

【范例 7-6】示例代码 7-6 是一个测试 final 常量的程序。

示例代码 7-6

```
01  public class StaticConstant {
02     public static void main(String[] args) {
03        StaticContantTest sct = new StaticContantTest();
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
04          System.out.println(sct.eat());
05      }
06  }
07
08  class StaticContantTest {
09      static final String str = "去吃饭...";
10
11      static String eat() {
12          return str;
13      }
14  }
```

【运行结果】程序运行结果如下。

```
去吃饭...
```

【代码解析】在程序中，首先创建一个 StaticContantTest 类，并在其中创建方法 eat()，设置一个字符串类型变量 str，使用 final 修饰。然后创建类 StaticConstant，在其中创建 StaticContantTest 的对象 sct，并调用 eat()的方法，打印输出 str 的值。

7.4.2　方法

如果一个方法被 final 修饰，那么这个方法就不可以被重写了，这提供了比较高的安全保证，但是，final 修饰方法应该小心使用，因为一旦使用了 final 关键字，那么被修饰的方法就会失去一些面向对象的优点。

【范例 7-7】示例代码 7-7 是一个测试 final 方法的程序。

示例代码 7-7

```
01  public class FinalMethod {
02      public static void main(String[] args) {
03          FinalTest ft = new FinalTest();
04          ft.a = 50;
05          ft.print();
06      }
07  }
08
09  class FinalTest {
10      int a;
11
12      final void print() {
13          System.out.println("a 的值是: " + a);
14      }
15  }
```

【运行结果】程序的运行结果如下。

```
a 的值是: 50
```

【代码解析】在以上的程序中，首先定义了一个类 FinalTest，并在该类中定义了成员变量 a 和一个 final 修饰的方法 print()，在方法中打印输出 a 的值，然后定义一个类 FinalMethod，并在主方法中创建 FinalTest 的对象 ft，并设置 a 的值，然后调用 print()方法，打印输出 a 的值。

7.4.3　类

如果一个类被 final 修饰，那么这个类就不能够再派生出子类，也就是说任何类都不能继承它。只有当该类中的所有方法都不需要重写的时候，才能将该类定义为 final 类。

【范例 7-8】示例代码 7-8 是一个测试 final 类的程序。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

示例代码 7-8

```
01  public class FinalClass {
02      public static void main(String[] args) {
03          FinalClassTest fct = new FinalClassTest();
04          fct.eat = "该去吃饭了";
05          fct.str = "十一点半";
06          fct.print();
07          fct.eat();
08      }
09  }
10
11  final class FinalClassTest {
12      String str;
13      String eat;
14
15      void print() {
16          System.out.println("现在时间是" + str);
17      }
18
19      void eat() {
20          System.out.println(eat);
21      }
22  }
```

【运行结果】程序运行结果如下。

```
现在时间是十一点半
该去吃饭了
```

【代码解析】在以上的程序中，首先定义了一个类 FinalClassTest，并在该类中定义了成员变量 str 和 eat，以及两个方法 print()和 eat()，在方法中打印输出 str 和 eat 的值，然后定义一个类 FinalClass，在主方法中创建 FinalClassTest 的对象 fct，并设置两个变量的值，最后调用 print()和 eat()方法，打印输出 str 和 eat 的值。



7.5 Object 类

Java 中，所有类的祖先类都是 Object，这是 Java 中存在的比较特殊的类。如果一个类没有继承任何的类，那么它默认是继承 Object 类的。如果它继承了一个类，那么它的父类或者父类的父类也继承于 Object 类。也可以说，任何类都是 Object 的子类。

7.5.1 Object 对象

Object 类比较特殊，在实际的项目开发中会经常用到。定义一个 Object 类型对象的数组，根据前面继承的知识，它可以放置任何的类型，可以把几个不同的对象放置到 Object 数组中，但是放置进去后，对象的类型就会被丢弃了，在取出后需要进行类型的转换。

7.5.2 重写 equals 方法和 toString 方法

如果要判断两个物体的长度是否相等，首先需要使用尺子将他们的长度都测量出来，然后再进行判断。Object 中也有一些类似于尺子的方法，其中比较重要的是 equals 和 toString 方法。Object 中对 equals 的定义如下。

```
public Boolean equals(Object obj){
        return(this==obj);
}
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

如果两个对象指向的是同一个对象，就会返回 true，也就是两个对象的地址相同，如果要进行详细的判断，需要对 equals 进行方法的重写，首先判断两个对象是否指向同一个对象，如果是的话，返回 true，如果不是则继续判断所提供的类型是否是字符串类型，如果不是则返回 false，如果是则进入下一步的判断，取出两个字符串的每一个对应位置的字符进行判断，如果有一个值不等，则返回 false。

toString 方法默认返回的是对象的地址，可以覆盖 toString 方法来实现用户所要输出的内容。

字符串是由字符组成的，Java 中的字符串是描述字符的基本数据结构，字符串具有不可变性，也就是 Java 中的字符串类型对象是不能被改变的。

【范例 7-9】示例代码 7-9 是一个测试重写 toString 的程序。

示例代码 7-9

```
01  public class ToStringTest {
02      public static void main(String[] args) {
03          ToString ts = new ToString();
04          ts.name = "张三";
05          ts.age = 22;
06          ts.sex = "男";
07          System.out.println(ts);
08      }
09  }
10
11  class ToString {
12      String name;
13      int age;
14      String sex;
15
16      public ToString() {
17      }
18
19      public ToString(String name, int age, String sex) {
20          super();
21          this.name = name;
22          this.age = age;
23          this.sex = sex;
24      }
25
26      public void setAge(int age) {
27          this.age = age;
28      }
29
30      public void setName(String name) {
31          this.name = name;
32      }
33
34      public void setSex(String sex) {
35          this.sex = sex;
36      }
37
38      public String toString() {
39          return "姓名:" + name + "\n年龄:" + age + "\n性别:" + sex;
40      }
41  }
```

【运行结果】程序运行结果如下。

```
姓名:张三
年龄:22
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

性别:男

【代码解析】在以上的程序中，首先定义了一个类 ToString，并创建了三个成员变量，然后设置有参构造方法和无参构造方法，并设置成员变量各自对应的 setter 方法。接下来重写 toString 方法，用于返回用户信息。创建类 ToStringTest，在主方法中创建对象 ts，并设置 ToString 中的成员变量值，最后打印输出 ts。



7.6 习题

【习题 7-1】写一个抽象类 Person，定义一个 Boolean 类型的 final 变量 liveForever，一个 final 方法 fly()，定义两个 abstract 方法——speak()和 move()。

【分析】该题主要考查抽象方法和 final 变量、方法。

【核心代码】

```
public abstract class Person {
   public final boolean liveForever = false;
   public final boolean fly() {
      return false;
   }
   public abstract String speak();
   public abstract String move();
}
```

【习题 7-2】写一个 Employee 类继承 Person，实现 speak()和 move()两个 abstract 方法，并定义一个 work()方法。

【分析】该题主要考查继承和实现抽象方法。

【核心代码】

```
public class Employee extends Person{
   public String work() {
      return "I have to work for my living.";
   }
   public String speak() {
      return "I am a employee.";
   }
   public String move() {
      return "I go to work by bus.";
   }
}
```

【习题 7-3】写一个 Employer 类继承 Person，实现 speak()和 move()两个 abstract 方法，实现 Manager 接口，此接口定义了一个 manage(Employee ee)方法。

【分析】该题主要考查实现接口和调用 final 方法。

【核心代码】

```
public interface Manager {
   String manage(Employee ee);
}

public class Employer extends Person implements Manager {
   public String manage(Employee ee) {
      if (ee.fly()) {
         return "You can fly, it's good.";
      } else {
         return "Oops, you can not fly.";
      }
   }
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
    public String speak() {
        return "I'm the boss.";
    }
    public String move() {
        return "I drive a car.";
    }
}
```

【习题 7-4】改写 Person 类，增加一个 final 属性 id，通过构造函数赋值。超类覆盖默认构造函数后，子类也要做相应的改变。

【分析】该题主要考查 final 属性的赋值。

【核心代码】

```
public abstract class Person {
    public final boolean liveForever = false;
    public final String id;

    public Person(String id) {
        this.id = id;
    }

    public final boolean fly() {
        return false;
    }

    public abstract String speak();
    public abstract String move();
}

    public Employee(String id) {
        super(id);
    }

    public Employer(String id) {
        super(id);
    }
```

【习题 7-5】改写 Person 类，重写 equals()和 toString()方法。

【分析】该题主要考查重写 equals()和 toString()方法。

【核心代码】

```
public abstract class Person {
    public final boolean liveForever = false;
    public final String id;
    public Person(String id) {
        this.id = id;
    }
    public final boolean fly() {
        return false;
    }
    public abstract String speak();
    public abstract String move();
    public boolean equals(Object p) {
        return p instanceof Person && this.id.equalsIgnoreCase(((Person) p).id);
    }
    public String toString() {
        return this.id;
    }
}
```

【习题 7-6】定义一个 Police 类，在其主方法中比较几个 Person 对象是否是同一个人。

【分析】该题主要考查 equals()和 toString()的用法。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

【核心代码】

```
public class Police {
    public static void main(String[] args) {
        Person p1 = new Employee("135");
        Person p2 = new Employee("246");
        Person p3 = new Employee("135");
        if (p1.equals(p2)) {
            System.out.println("p1 与 p2 是同一个人.");
        } else {
            System.out.println("p1 与 p2 不是同一个人");
        }
        if (p1.equals(p3)) {
            System.out.println("p1 与 p3 是同一个人");
        } else {
            System.out.println("p1 与 p3 不是同一个人");
        }
    }
}
```

【习题 7-7】自定义一个 CustomList 类，继承自 AbstractList。

【分析】该题主要考查类的继承。

【核心代码】

```
public class CustomList<T> extends AbstractList<T> {
    private List<T> list = new ArrayList<T>();
    public T get(int index) {
        return list.get(index);
    }

    public boolean add(T o) {
        return list.add(o);
    }

    public int size() {
        return list.size();
    }

    public static void main(String[] args) {
        CustomList<String> l = new CustomList<String>();
        l.add("test1");
        l.add("test2");
        for (String s : l) {
            System.out.println(s);
        }
    }
}
```

【习题 7-8】定义一个 final 类 Greed，包括两个方法 selfish()、miser()。

【分析】该题主要考查 final 类。

【核心代码】

```
public final class Greed {
    public void selfish() {
        System.out.println("Don't extend the selfish character.");
    }

    public void miser() {
        System.out.println("Don't be a miser.");
    }
}
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



【习题 7-9】定义一个类 School，要求实现多个接口，如 ImpartKnowledge、TrainAbility 等。

【分析】该题主要考查接口的实现。

【核心代码】

```
public interface ImpartKnowledge {
    void impart();
}

public interface TrainAbility {
    void train();
}

public class School implements ImpartKnowledge, TrainAbility {
    public void impart() {
        System.out.println("Imparting knowledge.");
    }

    public void train() {
        System.out.println("Trainning your abilities.");
    }
}
```

【习题 7-10】编写一个抽象类 SemifinishedHouse，实现一个方法 room()，定义一个抽象方法 fitment()。

【分析】该题主要考查抽象类的定义。

【核心代码】

```
public abstract class SemifinishedHouse {
    public void room() {
        System.out.println("There are three rooms.");
    }
    public abstract void fitment();
}
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第8章　多　　态

多态是面向对象编程三个重要的特性之一。在 Java 中，多态性可以通过综合应用来继承、覆盖，以及通过向上转型来实现。

本章首先从综合阐述面向对象编程的这些重要特征，引申到代码中的多态概念、多态带来的好处，以及多态能够解决的问题，然后通过实例详细讨论多态技术在编程中的应用。



8.1　多态的概述

描述人的行为的词汇“多面手”，描述某个产品功能的“多合一”，就是多态。在日常生活中，多态最典型的例子就是计算机的应用。

通过前面所介绍的继承关系，一个一无所有的人可以从他的父亲那里得到一些东西，如资产等。在继承了父亲所有的资产之后，就可以按照自己的方式管理这些资产，改变以前父亲的操作方式，这就叫“重写”；当然也可以使用与父亲相同的操作方式管理这些资产，但只要通过整合相关资源，就可以产生完全不同的成果，这就叫重载。

8.1.1　重写

一个类可以通过继承来产生一个新类，新的类继承了父类所有的成员变量和方法，并且在继承这些方法和变量的同时，也具有了自己特有的行为，如果父类中的某一些方法不能够满足新的要求，就可以在子类中修改从父类中所继承过来的方法，并可以同时定义自己的方法。

【范例 8-1】示例代码 8-1 是一个测试重写的程序。

示例代码 8-1

```
01  public class Fruit
02  {
03      void print()
04      {
05          System.out.println("这是父类的方法。");
06      }
07  }
08  class Apple extends Fruit
09  {
10      void print()
11      {
12          System.out.println("这是子类的方法。");
13      }
14  }
```

【代码解析】在以上程序中，首先定义了一个 Fruit 类，并在其中实现方法 print()，然后打印输出语句“这是父类的方法。”，接下来定义一个子类 Apple 并继承 Fruit 类，然后实现 print() 方法，并重写其中的输出语句，改为输出“这是子类方法。”

在子类继承父类后，在子类中经常都会修改父类中的方法。重写的目的在于，一方面修改父类中的方法，另一方面对父类的方法进行扩展，或者是在子类中定义具体的、特定的行为。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



【范例 8-2】示例代码 8-2 是一个测试重写的程序。

示例代码 8-2

```
01  public class SchoolTest {
02      public static void main(String[] args) {
03          School1 s1 = new School1("清华附中", 100080, 1000, "海淀区");
04          s1.print();
05      }
06  }
07
08  class School {
09      String sname;
10      int sid;
11      int snumber;
12
13      public School(String name, int id, int number) {
14          sname = name;
15          sid = id;
16          snumber = number;
17      }
18
19      void print() {
20          System.out.println("学校名是：" + sname + "，序号是：" + sid + "，人数为：
21  " + snumber);
22      }
23
24  }
25
26  class School1 extends School {
27      String sadder;
28
29      public School1(String name, int sid, int snumber, String adder) {
30          super(name, sid, snumber);
31          sadder = adder;
32
33      }
34
35      void print() {
36          System.out.println("学校名是：" + sname + "，序号是：" + sid + "，人数为：
37  " + snumber + "，地址是：" + sadder);
38      }
39  }
```

【运行结果】程序的运行结果如下。

学校名是：清华附中，序号是：100080，人数为：1000，地址是：海淀区

【代码解析】在以上程序中，首先定义了一个 School 类，并在其中声明了三个成员变量，并设置有参构造方法，然后新建一个 print()方法，用于打印输出学校的信息。接下来定义一个 School1 类，继承 School 类，在该类中声明一个成员变量，然后重写父类中的 print()方法，在打印输出语句中新加一个输出结果。最后创建 SchoolTest 测试类，创建 School1 的对象 s1，并调用 print()方法，打印输出结果。

父类中的方法不是在任何情况下都可以重写的，如果父类中的方法访问控制修饰符设置为了 private，该方法就只能被自己的类访问，不能被外部类访问。如果父类中的方法访问控制符设置为 public，而子类中该方法设置为了 private，则在编译的时候会报错。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

8.1.2 重载

前面介绍了重写，也许会有人认为两者类似，其实不然。重写和重载是两个完全不同的概念，重载是指在同一个类中有两个或者是更多个的方法可以使用一个相同的名称，不同的是它们的参数。只要参数不同，方法名相同是没有问题的，也就相当于在一个班级中有两个相同姓名的学生，名字虽然相同，但是长相是绝对不同的。

【范例 8-3】示例代码 8-3 是一个测试重载的程序。

示例代码 8-3

```
01  public class ClassTest
02  {
03      public static void main(String[] args)
04      {
05          Student stu=new Student();
06          stu.name="张三";
07          stu.age=20;
08          stu.print();
09          stu.print("李四", 21);
10          stu.print("王五", 22, 10);
11          stu.print("赵六", 22, 11, 80);
12
13      }
14  }
15  class Student
16  {
17      String name;
18      int age;
19
20      void print()
21      {
22          System.out.println("姓名："+name+"年龄："+age);
23      }
24
25      void print(String name1,int age1)
26      {
27          System.out.println("姓名："+name1+"年龄："+age1);
28      }
29
30      void print(String name2,int age2,int id)
31      {
32          System.out.println("姓名："+name2+"年龄："+age2+"学号是："+id);
33      }
34
35      void print(String name3,int age3,int id3,double a)
36      {
37          System.out.println("姓名："+name3+"年龄："+age3+"学号是："+id3+"成绩是：
"+a);
38
39      }
40  }
```

【运行结果】程序运行结果如下。

```
姓名：张三年龄：20
姓名：李四年龄：21
姓名：王五年龄：22 学号是：10
姓名：赵六年龄：22 学号是：11 成绩是：80.0
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



【代码解析】在以上程序中，首先定义一个 Student 类，在类中声明两个参数，分别代表学生的姓名和年龄，然后在该类中定义四个 print()方法，这四个方法分别设置了不同的参数，然后定义一个 ClassTest 类，作为测试类。创建 Student 类的对象 stu，并调用 print()方法，传入不同的参数打印输出。

在调用重载方法时，到底调用哪一个方法是基于其参数的。如果是 int 类型的参数调用该方法，则会调用带有 int 参数的重载方法。

有时候，会在同一段代码中遇到重写和重载方法，需要搞清楚重写是指在子类中修改父类中所继承的方法，并且在继承方法的同时可以定义自己的方法。而重载则是指在一个类中具有相同方法名而参数不同，它们必须具有相同的方法名。

【范例 8-4】示例代码 8-4 是一个测试重写和重载结合的程序。

示例代码 8-4

```
01  public class TwoTest {
02
03      public static void main(String[] args)
04      {
05
06          TestTwo tt=new TestTwo();
07          tt.a=5;
08          tt.b=6;
09          tt.sub();
10          tt.add();
11          tt.sub(5);
12
13      }
14
15  }
16  class Test
17  {
18      int a;
19      int b;
20      int add()
21      {
22          System.out.println("a 加 b 的值是："+a+b);
23          return a+b;
24      }
25      int sub()
26      {
27          System.out.println("a 减 b 的值是："+(a-b));
28          return a-b;
29      }
30  }
31  class TestTwo extends Test
32  {
33      int add()
34      {
35          System.out.println("a 的值是："+a);
36          return a;
37      }
38      int sub()
39      {
40          System.out.println("b 的值是："+b);
41          return b;
42      }
43      long sub(long a)
44      {
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
45            System.out.println("a 的值是："+(a));
46            return a;
47        }
48
49    }
```

【运行结果】程序运行结果如下。

```
b 的值是：6
a 的值是：5
a 的值是：5
```

【代码解析】在以上代码中，首先定义了一个 Test 类，并在该类中定义两个成员变量和两个方法，接着又定义了一个 TestTwo 类，该类继承 Test 类并重写了父类中的方法，同时 TestTwo 类也定义了自己的一个方法。最后创建了 TestTwo 类的一个对象，并使用该对象调用该类中的所有方法。TestTwo 类中的 add 方法和 sub 方法分别重写了父类中的 add 方法和 sub 方法，并且在 TestTwo 类中的 sub(a)重载了该类中的 sub 方法。

8.2 绑定

所谓绑定就是在对象引用与对象实例之间建立联系。绑定分为静态绑定和动态绑定。静态绑定又叫前期绑定，是在编译期建立的联系；动态绑定又叫后期绑定，是在运行期建立的联系。

8.2.1 静态绑定

静态绑定就是在编译时就决定了变量的值或应该调用哪个类的方法。由于这种联系是在编译时就建立的，所以其性能较好，可以提高代码的运行速度。Java 中属于静态绑定的有以下一些内容。

- 类属性。
- 构造方法。
- 声明为 static 的静态方法。
- 声明为 final 的方法，将其另列为一个项目，如下声明为 private 的方法。
- 使用关键字 super 调用的方法，包括 super()。

【范例 8-5】示例代码 8-5 是一个测试静态绑定的程序。

示例代码 8-5

```
01  public class Auto {
02      public String name = "an automobile";
03      public void engine() {
04          System.out.println("a poor engine");
05      }
06      public final void price() {
07          System.out.println("This is a budget machine.");
08      }
09      public static void run() {
10          System.out.println("a bit slow");
11      }
12      public void fun() {
13          System.out.println("Driving fun");
14      }
15  }
16  public class Car extends Auto {
17      public String name = "a nice car";
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
18      public void engine() {
19          System.out.println("a nice engine");
20      }
21      public static void run() {
22          System.out.println("fast");
23      }
24      public void fun() {
25          super.engine();
26      }
27      public static void main(String[] args) {
28          Auto auto = new Car();
29          System.out.println(auto.name);    // 类属性
30          auto.price();                     // 父类 final 方法
31          auto.run();                       // 父类静态方法
32          auto.fun();                       // super 关键字
33      }
34  }
```

【运行结果】程序运行结果如下。

```
an automobile
This is a budget machine.
a bit slow
a poor engine
```

【代码解析】在以上代码中，首先定义了一个 Auto 类，该类有一个类属性和四个方法，然后又定义了一个 Car 类，该类继承自 Auto，并重写了 engine()和 fun()方法，同时也定义了 name 类属性。在 main()方法中，auto 对象引用了一个 Car 实例，首先打印了类属性 auto.name，然后调用 final 方法 auto.price()及静态方法 auto.run()，在 auto.fun()方法中使用了 super 关键字调用了父类的 engine()方法，通过运行结果可以看出，这些都是静态绑定。

静态绑定的优点是能够提高程序的运行效率，所以在资源有限，又对程序性能要求比较高的地方，可以适当多用一些静态绑定。

8.2.2　动态绑定

动态绑定是指在运行时，根据对象引用所指向的具体实例决定执行什么样的方法，由于是在运行时由 JVM 完成的，因此会减慢程序运行速度。

动态绑定的好处是提高了程序设计的灵活度，降低了耦合性。

【范例 8-6】示例代码 8-6 是一个测试动态绑定的程序。

示例代码 8-6

```
01  public class Food {
02      public void taste() {
03          System.out.println("It's good.");
04      }
05
06      public void smell() {
07          System.out.println("Not bad");
08      }
09
10      public void feed() {
11          System.out.println("It's good.");
12      }
13  }
14  public class Noodles extends Food {
15      public void taste() {
16          System.out.println("It's delicious.");
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
17      }
18
19      public void smell() {
20          System.out.println("Smell sweet");
21      }
22
23      public void feed() {
24          System.out.println("It's nutritious.");
25      }
26
27      public static void main(String[] args) {
28          Food food = new Noodles();
29          food.taste();
30          food.smell();
31          food.feed();
32      }
33  }
```

【运行结果】程序运行结果如下。

```
It's delicious.
smell sweet
It's nutritious.
```

【代码解析】在以上代码中，首先定义了一个 Food 类，该类有三个方法：taste()、smell()和 feed()。然后又定义了一个 Noodles 类，该类继承自 Food，并重写了父类的三个方法。在 main()方法中，food 对象引用了一个 Noodles 的实例，并调用了 food 的三个方法，由于是动态绑定，实际调用的是引用实际指向的 Noodles 类的三个方法。

动态绑定为灵活程序设计提供了方便，在上面的例子中，如果将 food 对象的类型改为 Noodles，那么如果在需要一个米饭类（Rice）时就得重写代码了，定义了父类变量，在需要其他子类时，只要传递对象实例就可以了。



8.3 习题

【习题 8-1】写一个父类 Animal，定义 shout()和 move()方法，然后写一个 Dog 类，重写 Animal 的 shout()和 move()方法。

【分析】该题主要考查重写。

【核心代码】

```
public class Animal {
    public String shout() {
        return "shout...";
    }

    public String move() {
        return "walking like an animal.";
    }
}

public class Dog extends Animal {
    public String shout() {
        return "bark...";
    }

    public String move() {
        return "walking like a dog.";
    }
}
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

【习题 8-2】写一个类 Cat 继承自 Animal，重写 shout()和 move()方法，并重载 move()方法。

【分析】该题主要考查重写与重载。

【核心代码】

```
public class Cat extends Animal {
    public String shout() {
        return "mewing";
    }

    public String move() {
        return "walking like a cat.";
    }

    public String move(String tree) {
        return "climbing a tree " + tree;
    }
}
```

【习题 8-3】给 Animal 类定义一个公共的 form 属性，然后定义一个 final 方法 friend()和一个 static 方法 protect()。写一个 Person 类，在其 main()方法中创建一个【习题 8-1】中的 Dog 对象传递给 Animal 引用，然后调用此引用的 from 属性、friend()和 protect()方法。

【分析】该题主要考查静态绑定。

【核心代码】

```
    public String form = "wild-animals";

    public final String friend() {
        return "Animals are our friends.";
    }

    public static String protect() {
        return "Protect animals.";
    }

public class Person {
    public static void main(String[] args) {
        Animal animal = new Dog();
        System.out.println(animal.form);
        System.out.println(animal.friend());
        System.out.println(animal.protect());
    }
}
```

【习题 8-4】在 Person 类的 main()方法中新建一个 Cat 对象实例传递给 animal 引用，然后调用其 shout()和 move()方法。

【分析】该题主要考查动态绑定。

【核心代码】

```
animal = new Cat();
System.out.println(animal.shout());
System.out.println(animal.move());
```

【习题 8-5】自定义一个类 Tree，重写 Object 的 equals 和 toString 方法。

【分析】该题主要考查方法的重写。

【核心代码】

```
public class Tree extends Object {
    private String kind;
    private String position;
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
    public Tree(String kind, String position) {
        this.kind = kind;
        this.position = position;
    }

    public boolean equals(Object obj) {
        if (null == obj) return false;
        Tree o = (Tree) obj;
         return (kind.equals(o.kind) && position.equals(o.position));
    }

    public String toString() {
         return kind + "@" +position;
    }
}
```

【习题 8-6】自定义一个超类 Book，然后继承 Book 实现一个 JavaBook 类，在构造方法中调用超类的构造方法，重写其方法。

【分析】该题主要考查方法的重写。

【核心代码】

```
public class Book {
    public String title;
    public String pages;

    public Book(String title, String pages) {
        this.title = title;
        this.pages = pages;
    }

    public void record() {
        System.out.println("Record knowledge");
    }
}

public class JavaBook extends Book {
    public JavaBook(String title, String pages) {
        super(title, pages);
    }

    public void record() {
        System.out.println("Record java knowledge.");
    }
}
```

【习题 8-7】编写一个类 StringTool，这个类的功能是将 Integer、Person、Book 等类型的对象转换成字符串。

【分析】该题主要考查方法的重载。

【核心代码】

```
public class StringTool {
    public String beString(Integer x) {
        return x.toString();
    }

    public String beString(Person p) {
        return p.toString();
    }

    public String beString(Book b) {
        return b.title + " : " + b.pages;
    }
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
}
```

【习题 8-8】自定义一个类 NewList 继承自 ArrayList，重写其方法，并使用 String 类型作为参数重载其 get 方法。

【分析】该题主要考查重写和重载。

【核心代码】

```
public class NewList<T> extends ArrayList<T> {
    public boolean add(T t) {
        return super.add(t);
    }

    public T get(int i) {
        return super.get(i);
    }

    public T get(String i) {
        return get(Integer.parseInt(i));
    }

    public int size() {
         return super.size();
    }
}
```

【习题 8-9】自定义一个类 Mouse，定义一个 final 属性，一个 static 方法，一个 final 方法。

【分析】该题主要考查定义方法和属性。

【核心代码】

```
public class Mouse {
    public final String kind = "mammals";

    public final void steal() {
        System.out.println("need more");
    }
    public static void dig() {
        System.out.println("dig a hole");
    }
}
```

【习题 8-10】编写一个仓鼠类 Hamster，继承第 9 题的 Mouse 类。

【分析】该题主要考查类的继承。

【核心代码】

```
public class Hamster extends Mouse{
    public void store() {
        System.out.println("store food");
    }

    public void pamper() {
        System.out.println("a pet hamster");
    }
}
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第 9 章　接　　口

接口是 Java 语言中为解决无法实现多继承而提供的一个解决方案。接口中并没有实际的方法实现，只定义了一些空壳，一个类可以通过实现接口来实现多重继承关系。本章主要介绍接口的定义及使用，以及内部类和包的概念。



9.1　认识接口

接口在 Java 中是一项比较重要的技术，它就相当于一个构建出来的模型，只有一个空壳，而实际要建造的物体必须要依照这个模型的一些属性，实现具体的功能。它是一系列抽象方法和常量的属性集合，一个类如果实现了接口，就必须要实现接口中所定义的所有方法，而且一个类可以实现多个接口。实际上，接口只是一个类的框架，是一种完全形式上的抽象类。

9.1.1　概述

前面介绍了抽象类，而接口就是一种特殊的抽象类。接口中只提供了方法的原形，但没有具体的方法实现。子类必须提供父类中的抽象方法的实现方式。接口和抽象类一样，也可以定义抽象的方法，并且在接口中只能够包含抽象方法和常量，不能够有变量、初始化块和构造函数。这就好比一辆汽车模型，这个模型具有和实际汽车同样的大小和外观，但是这只是一个模型，并不能发动，如果想要实现发动功能的话，就需要在其中安装一系列的功能配件，也就是具体的方法实现。

创建一个接口，那么这个接口中的方法必须是抽象方法，也就是其中不能提供具体的方法实现，而是只定义要做什么，而不需要去实现怎么做。定义一个接口之后，任何的类都可以去实现这个接口，而且一个类可以去实现多个接口。接口需要使用关键字 interface 来进行修饰。

9.1.2　定义接口

接口的定义跟类的定义十分相似，它的语法如下。

```
修饰符 interface 接口名
{//接口内容
 //声明变量
 数据类型 变量名;
 抽象方法;
 ……
//声明方法
 返回值类型 方法名();
  ……
}
```

在定义接口的时候，需要注意以下几点。

- 设置接口的修饰符，接口的修饰符只有两种选择，即默认无修饰符和 public 两种。如果没有修饰符，就说明接口对于所在包内的类是可见的，接口所在的包之外的类不可以使用该接口。如果修饰符是 public，则说明任何的类都可以使用该接口。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



- 接口的名称设定需要满足 Java 标识符的规定。
- 接口定义的方法都为抽象方法，它们被自动地设置为 public 的。

【范例 9-1】示例代码 9-1 是一个定义接口的程序。

示例代码 9-1

```
01  public interface Student
02  {
03  //接口中的变量
04  int Class=0604;          //设置班级
05  //用接口声明方法，只有方法的声明没有具体实现
06  void Learning ();        //设置学习方法
07  void eat();              //吃饭方法
08  void sleep();            //睡觉方法
09  }
```

【代码解析】在以上的代码中，声明了一个整型的变量（第 04 行），在该变量声明时，只说明是一个整型。由于接口的特殊性，该变量会被认为是公共的、静态的、最终的字段，而下面所定义的三个方法（第 06～08 行），也会全部默认被定义为公共字段。

9.1.3　实现接口

设计完成汽车模型之后，接下来就是根据模型及所规定的方法来造一辆真正的汽车，首先需要按照模型建造外观，然后需要实现模型中的方向盘、油门、刹车等一系列的功能。

所谓接口的实现，就是实现接口的类。在接口声明的时候，只给出了抽象方法，也就是打好了框架。实现接口需要实现接口中所定义的所有方法。接口的实现是指具体实现接口的类。接口的声明仅仅给出了抽象方法，相当于事先定义了程序的框架。实现接口的类必须要实现接口中定义的方法。接口实现的格式如下。

```
class 类名 implements 接口 1,接口 2
{
  方法 1 () {
  //方法体
}
方法 2 () {
//方法体
}
}
```

由关键字表示实现的接口，多个接口之间用逗号隔开。实现接口需要注意以下几点。

- 为接口中的所有方法提供具体的实现。
- 必须遵守重写的所有规则。
- 保持相同的返回类型。

【范例 9-2】示例代码 9-2 是一个接口的实现程序。

示例代码 9-2

```
01  public class StudentTest {
02      public static void main(String[] args) {
03          Student1 stu=new Student1();
04          stu.Learning();
05          stu.eat();
06          stu.sleep();
07      }
08  }
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
09  interface Student {
10      // 接口中的变量
11      int Class = 0604; // 设置班级
12      // 用接口声明方法，只有方法的声明，没有具体实现
13      void Learning();  // 设置学习方法
14      void eat();       // 吃饭方法
15      void sleep();     //睡觉方法
16  }
17  class Student1 implements Student
18  {
19
20      public void Learning() {
21          // TODO Auto-generated method stub
22          System.out.println("学生正在学习..");
23      }
24      public void eat() {
25          // TODO Auto-generated method stub
26          System.out.println("学生正在吃饭...");
27      }
28      public void sleep() {
29          // TODO Auto-generated method stub
30          System.out.println("学生正在睡觉...");
31      }
32  }
```

【运行结果】程序的运行结果如下：

```
学生正在学习..
学生正在吃饭...
学生正在睡觉...
```

【代码解析】以上程序中，首先定义了一个接口 Student（第 09 行），其中定义了 learning()、eat()、sleep()三个方法（第 12～15 行），然后创建类 Student1，并实现接口 Student，实现其中的三个方法，分别打印输出（第 17～30 行），最后在测试类 StudentTest 中进行调用。

9.1.4 多重实现接口

如果汽车不仅有一个外观的模型，而且里面的各个部件也都有相应的模型，如方向盘的样式、座椅样式等，就叫做使用了多个模型。实现每一个部件的时候，都需要按照一个不同的模型来制造和实现。

如果一个类实现了多个接口，那么这些接口之间就需要使用逗号来隔开。如果这几个接口都有相同的方法和相同的常量，那么相同的常量可以通过<接口名>.<常量名>的形式来访问，相同的方法将被其中的一个接口使用。

【范例 9-3】示例代码 9-3 是一个多重实现接口的程序。

示例代码 9-3

```
01  public class OperationTest {
02
03      public static void main(String[] args) {
04          OTest ot=new OTest();
05          System.out.println("a+b 的值是: "+ot.add(15,3));
06          System.out.println("a-b 的值是: "+ot.sub(15, 3));
07          System.out.println("a*b 的值是: "+ot.mul(15, 3));
08          System.out.println("a/b 的值是: "+ot.umul(15, 3));
09      }
10
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
11  }
12  interface Oadd
13  {
14      int add(int a,int b);
15  }
16  interface Osub
17  {
18      int sub(int a,int b);
19  }
20  interface Omul
21  {
22      int mul(int a,int b);
23  }
24  interface Oumul
25  {
26      int umul(int a,int b);
27  }
28  class OTest implements Oadd,Osub,Omul,Oumul
29  {
30
31      public int add(int a, int b) {
32          // add 方法的具体实现
33          return a+b;
34      }
35
36      public int sub(int a, int b) {
37          // sub 方法的具体实现
38          return a-b;
39      }
40
41      public int mul(int a, int b) {
42          // mul 方法的具体实现
43          return a*b;
44      }
45
46      public int umul(int a, int b) {
47          // umul 方法的具体实现
48          return a/b;
49      }
50
51  }
```

【运行结果】程序运行结果如下。

```
a-b 的值是：12
a*b 的值是：45
a/b 的值是：5
```

【代码解析】在以上的程序中，首先创建了四个接口，每一个接口代表一种基本运算（第12～27行）。然后创建了 OTest 类，分别继承四个接口，并在类中实现接口中所定义的方法（第28～49行），也就是返回各个运算的值。最后在测试类中创建 OTest 的对象，并分别打印输出各个运算的值。

9.1.5 接口的引用

在 Java 中，也可以建立接口类型的引用变量。接口的引用变量可以存储一个指向对象的值引用，而这个对象可以是任何的实现该接口的实例。对象的方法必须是类中的非抽象方法。

【范例 9-4】示例代码 9-4 是一个实现接口引用的程序。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

示例代码 9-4

```
interface ladd
{
    int add(int a,int b);
}
interface lsub
{
    int sub(int a,int b);
}
interface lmul
{
    int mul(int a,int b);
}
interface lumul
{
    int umul(int a,int b);
}
class Algor implements ladd,lsub,lmul,lumul
{

    public int add(int a, int b) {
        return a+b;
    }

    public int sub(int a, int b) {
        return a-b;
    }

    public int mul(int a, int b) {
        return a*b;
    }

    public int umul(int a, int b) {
        return a/b;
    }

}
public class AlgorTest {
    public static void main(String[] args) {
        Algor al=new Algor();
        ladd ad=al;
        lsub su=al;
        lmul lm=al;
        lumul lu=al;
        System.out.println("这是使用对象引用来调用方法：");
        System.out.println("a+b 的值是："+al.add(6, 3));
        System.out.println("a-b 的值是："+al.sub(6, 3));
        System.out.println("a*b 的值是："+al.mul(6, 3));
        System.out.println("a/b 的值是："+al.umul(6, 3));
        System.out.println("下面这是使用接口引用调用方法：");
        System.out.println("a+b 的值是："+ad.add(9, 3));
        System.out.println("a-b 的值是："+su.sub(9, 3));
        System.out.println("a*b 的值是："+lm.mul(9, 3));
        System.out.println("a/b 的值是："+lu.umul(9, 3));
    }

}
```

【运行结果】程序运行结果如下。

这是使用对象引用来调用方法：

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
a+b 的值是：9
a-b 的值是：3
a*b 的值是：18
a/b 的值是：2
下面这是使用接口引用调用方法：
a+b 的值是：12
a-b 的值是：6
a*b 的值是：27
a/b 的值是：3
```

【代码解析】在以上的代码中，首先定义了几个简单算法的接口（第 01～16 行），然后让类 Algor 实现了几个接口中的方法（第 17～33 行）。在测试类中声明各个接口的引用，并分别使用对象引用调用方法和接口引用调用方法来打印输出各个算法的值（第 37～53 行），并做比较。

9.2　内部类

所谓内部类，就是定义于类内部的类，这就好比把人看做是一个类，人的身体中又有不同的器官，这时可以把每一个器官再看做是一个类，器官中的不同组织完成不同的功能，也就是不同的方法实现。

内部类是一种非常有用的机制，通过内部类的定义，程序员将一些逻辑上相关的类组织在一起，并且通过内部类的使用可以很好地控制内部类的可见性。

9.2.1　内部类概述

在 Java 中，一个类被嵌套定义在另一个类中，那么这个类就被称为是内部类，包含内部类的类就被称为外部类。对于外部类来说，内部类相当于成员变量或者是方法。创建一个内部类和创建一个普通的类没有多大的区别，唯一不同的就是内部类是定义在类的内部。

【范例 9-5】示例代码 9-5 是一个定义内部类的程序。

示例代码 9-5

```
01  class Outer
02  {
03      String out_string="String in Outer";    //设置外部类的变量
04      void useIner(){                         //设置外部类的方法
05          Iner in=new Iner();                 //在外部类中对内部类的对象进行定义
06          in.print();                         //调用内部类的方法
07      }
08      //创建内部类
09      class Iner
10  {
11          void print(){                       //在内部类中定义的方法
12              System.out.println("Iner.print()");
13              //在内部类中访问其宿主类的变量
14              System.out.println("use\'"+out_string+"\'");
15          }
16      }
17  }
18  public class InnerTest
19  {
20      public static void main(String[] args)
21      {
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
22            Outer out=new Outer();          //创建一个外部类对象
23            out.useIner();                  //调用该类的内部类定义的方法
24        }
25    }
```

【运行结果】程序的运行结果如下。

```
Iner.print()
use'String in Outer'
```

【代码解析】在以上示例中，在 Outer 类的内部定义了一个内部类 Iner（第 09～14 行），并在 Outer 的方法 useIner 中生成一个内部类的对象（第 04～06 行），通过这个对象调用内部类的方法 print。

9.2.2 匿名内部类

所谓匿名内部类，就是没有名称的内部类。

【范例 9-6】示例代码 9-6 是一个定义匿名内部类的程序。

示例代码 9-6

```
01  public class AnonymousInnerTest {
02      public static void main(String[] args) {
03          DustMan dm=new DustMan();
04          dm.cleanWithBroom();
05          dm.cleanWithDustCollector();
06      }
07
08  }
09  class DustMan
10  {
11      public void cleanWithBroom(){
12          Utility util =new Utility(){
13              public void clean(){
14                  System.out.println("正在清洁卫生...");
15              }
16          };
17          util.clean();
18      }
19      public void cleanWithDustCollector(){
20          ElectronicUtil util=new ElectronicUtil(){
21              public void clean(){
22                  System.out.println("正在使用吸尘器清洁卫生....");
23              }
24          };
25          util.clean();
26      }
27  }
28  interface Utility{
29      public void clean();
30  }
31  class ElectronicUtil{
32      public void clean(){
33          System.out.println("使用电子设备清洁卫生...");
34      }
35
36  }
```

【运行结果】程序运行结果如下：

```
正在清洁卫生...
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

正在使用吸尘器清洁卫生....

【代码解析】在以上示例中，建立了一个匿名内部类，这个类实现了 Utility 接口并且重写了 Utility 接口中的 clean 方法（第 21 行和第 22 行）。由于建立这个类的时候没有类名，所以称之为匿名内部类。

如果匿名类所继承或实现的接口，只有带参数的构造方法，没有无参数构造方法，该用如下形式创建此匿名类：

```
父类名 对象名=new 父类名（参数列）
{
具体实现代码;
}
```

【范例 9-7】示例代码 9-7 是一段创建带有参数的匿名内部类的程序。

示例代码 9-7

```
01  public class SporterTest {
02      public void drink(Beverages drink){
03          System.out.println("运动员在引用"+drink);
04      }
05      public static void main(String[] args) {
06          SporterTest sp=new SporterTest();
07          sp.drink(new Beverages("橙汁"){
08              public void addVataminc(){
09                  System.out.println("添加维他命 C");
10              }
11          });
12
13          sp.drink(new Beverages("可乐"){
14              public void addVataminc(){
15                  System.out.println("添加二氧化碳");
16              }
17          });
18
19          sp.drink(new Beverages("雪碧"){
20              public void addVataminc(){
21                  System.out.println("添加二氧化碳");
22              }
23          });
24
25          sp.drink(new Beverages("芬达"){
26              public void addVataminc(){
27                  System.out.println("添加色素");
28              }
29          });
30          sp.drink(new Beverages("绿茶"){
31              public void addVataminc(){
32                  System.out.println("添加色素");
33              }
34          });
35      }
36
37  }
38  class Beverages
39  {
40      private String juiceName;
41      public Beverages(String name)
42      {
43          this.juiceName=name;
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
44        }
45        public String toString()
46        {
47        return juiceName;
48        }
49    }
```

【运行结果】程序运行结果如下。

```
运动员在引用橙汁
运动员在引用可乐
运动员在引用雪碧
运动员在引用芬达
运动员在引用绿茶
```

【代码解析】在以上示例中，Beverages 类表示的是饮料（第 38～47 行），而 SporterTest 则是代表着运动员的类，运动员需要喝饮料，所以就需要有一个 drink 方法（第 02～04 行）。在主方法中临时建立了带有参数的匿名内部类。

9.2.3 继承内部类

前面介绍了引用外部类的对象，内部类需要用到外部类的引用，就需要继承一个内部类，这样，就必须要解决内部类所对应的外部类引用初始化的问题。

【范例 9-8】示例代码 9-8 是一段关于继承内部类的程序示例。

示例代码 9-8

```
01  public class InnerClassTest {
02
03      class Clothes{
04          private String material;                    //设置材料
05          private int size;                           //设置大小
06          private String color;                       //设置颜色
07          public String getColor() {                  //创建 set/get 方法
08              return color;
09          }
10          public void setColor(String color) {
11              this.color = color;
12          }
13          public String getMaterial() {
14              return material;
15          }
16          public void setMaterial(String material) {
17              this.material = material;
18          }
19          public int getSize() {
20              return size;
21          }
22          public void setSize(int size) {
23              this.size = size;
24          }
25      }
26
27  }
28  class Coat extends InnerClassTest.Clothes{
29      public Coat (InnerClassTest o){
30          o.super();              //初始化外部类的引用
31      }
32  }
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

【代码解析】以上程序中的 Coat 类继承了内部类 Clothes（第 28 行），在构造方法中，o.super() 实现了初始化父类引用的功能（第 29～30 行）。

9.2.4 静态内部类

一般情况下，内部类会隐性持有一个指向外部类的引用，由此就可以直接去访问外部类的一些成员，如果不需要让内部类持有这个特性的话，就需要将内部类设置为静态属性。

【范例 9-9】示例代码 9-9 是一个静态内部类示例。

示例代码 9-9

```
01  public class StaticInnerClass {
02      public static void main(String[] args) {
03          Painter p=new Painter();
04          p.setName("张三");
05          p.print();
06      }
07
08  }
09  class Painter{
10      private Pen pen=new Pen();
11      private String name="";
12      static class Pen{
13          private String lead="铅笔芯";
14          public void print(){
15              System.out.println("使用铅笔..");
16          }
17      }
18      public String getName() {
19          return name;
20      }
21      public void setName(String name) {
22          this.name = name;
23      }
24      public Pen getPen() {
25          return pen;
26      }
27      public void setPen(Pen pen) {
28          this.pen = pen;
29      }
30      public void print(){
31          pen.print();
32      }
33  }
```

【运行结果】程序运行结果如下：

```
使用铅笔..
```

【代码解析】在以上程序中，将 Pen 类设置成了静态类型，所以不能够访问到外部类 Painter 的成员 name，以及它所对应的 setter 和 getter 方法。

静态内部类和普通内部类的区别是，普通内部类的变量和方法不能够被定义成静态内部类，也不能包含静态内部类，但是静态内部类却可以包含这些元素。

9.2.5 引用外部类的对象

如果需要在内部类中引用外部类的 this 指针，需要使用“方法名.This”的格式，如下所示。

```
print().this
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

【范例 9-10】示例代码 9-10 是一个使用引用外部类对象的示例。

示例代码 9-10

```
01  public class OutObjectTest {
02      public static void main(String[] args) {
03          Sun sun=new Sun();
04          Sun.H2 h2=sun.getH2();
05          h2.burn();
06      }
07
08  }
09
10  class Sun{
11      private int temperature=3600;
12      private String type="star";
13      private H2 fuel=null;
14      class H2{
15          private String name="氢气";
16          public String toString(){
17              return name;
18          }
19          public void burn(){
20              System.out.println(Sun.this+"正在燃烧"+this);
21          }
22      }
23      public String toString(){
24          return "太阳";
25      }
26      public H2 getFuel() {
27          return fuel;
28      }
29      public void setFuel(H2 fuel) {
30          this.fuel = fuel;
31      }
32      public int getTemperature() {
33          return temperature;
34      }
35      public void setTemperature(int temperature) {
36          this.temperature = temperature;
37      }
38      public String getType() {
39          return type;
40      }
41      public void setType(String type) {
42          this.type = type;
43      }
44      public H2 getH2(){
45          if(fuel == null){
46              fuel=new H2();
47          }
48          return fuel;
49      }
50
51  }
```

【运行结果】程序运行结果如下。

```
太阳正在燃烧氢气
```

【代码解析】在以上示例中，类 H2 是 Sun 的内部类（第 14～22 行），在 H2 的方法中，调用了外部类 Sun 中的 this 指针，所以下面的输出最终调用的是 Sun 类中的 toString 方法。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

9.3　包

包是 Java 面向对象语言中特有的概念，包将相关的一组类组织在一起进行管理，相当于操作系统中的资源管理器。

9.3.1　创建包

创建一个包的目的是为了防止创建重名的类，这样就可以避免编译器无法识别当前调用的是哪一个类的问题。例如，在学校中找一位名叫“张三”的学生，可是这个学校中存在有好几位“张三”，这时，如果直接找“几年级几班的张三”，那就容易得多了，避免了一些不必要的麻烦。

Java 中的包有命名包和未命名包两种。

使用未命名包的源文件，会默认操作系统中当前文件夹作为包，而且这种未命名包都处于顶层。在同一个源文件中的所有类都默认属于同一个包，如果这些类不在同一个源文件夹中，但是都使用的是未命名包，又出于同一个工作目录中，那么也会认为这些类属于同一个包中。

未命名包在使用上非常简单，但是这样的话就会发生命名的冲突，而且非常不便于对类进行管理。命名包的创建非常简单，只需要在类的第一行写上 package 语句就可以了，示例如下。

```
package 包名;
```

【范例 9-11】示例代码 9-11 是一个使用包的 HelloWorld 示例。

示例代码 9-11

```
01  package org.gaozou.book.javaabc.examples.ch9;
02
03  public class HelloWorld
04  {
05      public static void main(String args[]){
06          System.out.println("HelloWorld!");
07      }
08  }
```

【代码解析】在以上程序中，第一行写了一个语句“package org.gaozou.book.javaabc.examples.ch9;”，这就是在创建一个名为 com.cn.ch09 的包。

如果使用 MyEclipse 进行项目开发，那么创建包就更加容易了。方法是：右击项目名，选择“New”→“Package”菜单项，弹出“New Java Package”对话框，如图 9-1 所示，在“Name”文本框中输入包名，单击“Finish”按钮即可完成创建。

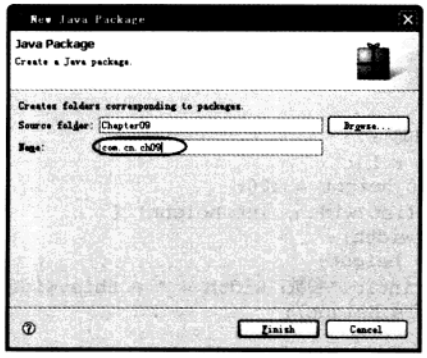


图 9-1　新建包

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

9.3.2 导入包

由于不同的包之间的类不可以直接相互引用，因此就必须要通过导入包的方式进行。包的导入有两种方式，一种是临时的导入，也就是在使用到该类的时候，直接输入完整的包名；另一种是通过 import 关键字来实现。使用 import 关键字的格式如下。

```
import 包名.类名;
```

如果需要引入该包中的所有的类，方法如下。

```
import 包名.*;
```

【范例 9-12】示例代码 9-12 是一个导入包的示例。

示例代码 9-12

```
01  package org.gaozou.book.javaabc.examples.ch9;
02  import java.io.File;
03  public class ImportTest {
04
05      public static void main(String[] args) {
06
07          File file=new File("test.java");
08
09          java.util.Date date=new java.util.Date();
10
11          System.out.println("当前时间是："+date);
12
13          java.math.BigDecimal decimal=new java.math.BigDecimal(5);
14
15          System.out.println("decimal:"+decimal);
16      }
17
18  }
```

【代码解析】在以上程序中，分别使用了两种不同的导入包方式，其中第 02 行是使用 import 关键字的形式导入的包，而后面的 Date 和 BigDecimal 均是使用临时导入的方式，直接输入完整的包名，完成各自相应的功能。

9.4 习题

【习题 9-1】编写一个程序，首先要定义一个外部类，在这个类里还要定义一个内部类 InnerClass 类，这里的 InnerClass 类是一个成员内部类。

【分析】该题主要考查的是内部类的使用。

【核心代码】

```
class OuterClass {
    String var = "外部类";
    private int width = 10;
    private static int height = 100;
    public OuterClass(int width, int height) {          // 带参数的构造方法
        this.width = width;
        this.height = height;
        System.out.println("参数：width = " + this.width + ",height = "
                + this.height);
    }
    private void outerOne() {                           // 外部类非静态方法
        System.out.println("系统正在访问外部类的非静态方法");
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
        InnerClass inner = this.new InnerClass();          // 外部类创建内部类对象
    }
    private static void outerTwo() {                       // 外部类的静态方法
        System.out.println("系统正在访问外部类的静态方法");
    }
    class InnerClass {                                     // 定义成员内部类
        String var = "内部类";
        public InnerClass() {                              // 默认构造方法
            System.out.println("创建了一个内部类对象");
        }
        protected void showVarValue() {          // 内部类调用外部类的 private 变量
            System.out.println("外部类的 width 变量值为: " + width);
            System.out.println("外部类的 height 变量值为: " + height);
        }
        protected void innerMethod() {           // 成员内部类的非静态方法
            System.out.println("允许内部类访问外部类的 private 方法");
            outerOne();
            outerTwo();
        }
    }
```

【习题 9-2】编写一个内部类程序，首先要定义一个抽象类，而且在该抽象类中定义三个抽象方法，要在外部类中可以通过方法来实现抽象类中的局部内部类，在该内部类中实现一个result()方法，并在该方法中定义一个局部内部类实现抽象类中的三个方法。

【分析】该题主要考查的局部内部类的使用及引用外部类的对象。

【核心代码】

```
abstract class Local {                                 // 定义一个抽象类
    public abstract void A_local();                    // 定义三个抽象方法
    public abstract void B_local();
    public abstract void C_local();
}
class OuterClass_1 {                                   // 外部类
    private String str = "现在访问的是外部类";
    private void p_showMess() {
        System.out.println("现在访问的是外部类的私有非静态变量——p_showMess()");
    }
    public void showMess() {
        System.out.println("现在访问的是外部类的非静态方法——showMess()");
    }
    public Local result() {                    // 方法返回实现的抽象类的局部内部类
        final String inner = "局部内部类是在方法中创建的类";
        class InnerClass_1 extends Local {// 定义局部内部类并继承抽象类，实现抽象方法
            public void A_local() {
                showMess();
            }
            public void B_local() {
                p_showMess();
            }
            public void C_local() {
            System.out.println("显示局部内部类的成员常量 inner: \"" + inner + "\"");
                System.out.println("调用外部类的成员变量 str: \"" + str + "\"");
            }
        }
        return new InnerClass_1();
    }
```

【习题 9-3】编写一个程序，首先创建一个植物类，在植物类中创建一个名为水果的内部类。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

创建一个苹果类，该类继承自水果类。在苹果类的主方法中，创建一个苹果对象，要求创建苹果对象的同时调用父类的构造方法。

【分析】该题主要考查内部类的使用，主要是继承内部类的使用。

【核心代码】

```
public class Plant {
    public String shape ="椭圆型";                    //叶子的形状
    private static String color = "白色";             //花的颜色
    private String grow="黑土地";                     //生长的环境
    public static void main(String[] args) {
        Fruit f = new Fruit();                        //创建 Fruit 类
        f.print();                                    //调用 Fruit 类的 print 方法
        new Apple().main(args);                       //调用 Apple 类 main 方法
    }
    static class Fruit{
        String taste ;                                //味道
        boolean iron;                                 //是否有果核
        Fruit(){                                      //给变量赋初值
            this.taste = "酸";
            this.iron =true;
        }
        public void print(){
            //调用外部类的变量
            System.out.println("叶子的形状："+new Plant().shape);
            System.out.println("花的颜色："+color);
            System.out.println("生长的环境："+new Plant().grow);
        }
    }
}
public class Apple extends Plant.Fruit {
    Apple(String taste) {                             // 自定义带参构造方法
        this.taste = taste;                           // 重新给 taste 变量赋值
    }
    Apple() {
        super();                                      // 调用父类的构造方法
    }
    public static void main(String[] args) {
        Apple apple = new Apple();                    // 创建 Apple 对象
        // 调用父类的变量
        System.out.println("水果的味道：" + apple.taste);
        System.out.println("是否有核：" + apple.iron);
    }
}
```

【习题 9-4】求出 n 的倍数与幂数。

【分析】该题主要考查的是接口的定义及实现。该题以数字 5 为例，求出它的幂数与倍数。先算出 5 的倍数，其实这个理解起来很简单，也就是对 5 这个数不断地加 5 操作就可以，即 n 的倍数公式为：n 的倍数=n+n+…+n；而 5 的幂数就是多个 5 相乘得出的结果，因此推出 n 的幂数公式为：n 的幂数=n*n*…*n。

【核心代码】

```
interface Number {
    int getNextVal();                                 // 获取下一个返回值
    void setInitVal();                                // 初始化变量 start、val
    void setMult(int x);                              // 设置某点的数值，即给 val 重新赋值
}
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
class PowerOfFive implements Number {          // 求5的n次方
    int start;                                 // 进行运算的数字
    int val;                                   // 结果
    PowerOfFive() {
        start = 1;
        val = 1;
    }
    // 实现接口中定义的所有方法
    public int getNextVal() {
        val *= 5;
        return val;
    }
    public void setInitVal() {
        start = 1;
        val = 1;
    }
    public void setMult(int n) {
        start = n;
        val = n;
    }
}
class MultOfFive implements Number {           // 求5的n倍的值
    int start;
    int val;
    MultOfFive() {
        start = 1;
        val = 1;
    }
    //实现接口中定义的所有方法
    public int getNextVal() {
        setMult(val);                          // 调用setMult方法，重新设置运算数
        val *= 5;
        return val;
    }
    public void setInitVal() {
        start = 1;
        val = 1;
    }
    public void setMult(int n) {
        start = n;
        val = n;
    }
}
```

【习题9-5】编写一个公司类，在该类中定义两个内部类，分别实现开发部和销售部的功能。公司产品的开发与销售工作都交给这两个内部类完成。

【分析】此题主要考查内部类的使用。

【核心代码】

```
public class Corp {

   private DevDep devDep = new DevDep();
   private SaleDep saleDep = new SaleDep();

   public String develop() {
      return devDep.work();
   }

   public String sale() {
      return saleDep.work();
   }
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
    class DevDep {
        public String work() {
            return "开发部正在工作...";
        }
    }

    class SaleDep {
        public String work() {
            return "销售部正在工作...";
        }
    }
}
```

【习题 9-6】在第 5 题的程序中定义一个 Department 接口，定义 work 方法，公司的每个部门必须实现 Department 接口。

【分析】此题主要考查接口与内部类的使用。

【核心代码】

```
public interface Department {
    String work();
}

    class DevDep implements Department {
        public String work() {
            return "开发部正在工作...";
        }
    }

    class SaleDep implements Department {
        public String work() {
            return "销售部正在工作...";
        }
    }
```

【习题 9-7】第 5 题中的 Corp 类实现 Organization 接口、Business 接口和 Company 接口。

【分析】此题主要考查接口的多重实现。

【核心代码】

```
public interface Organization {
    String goal();
}

public interface Business {
    String makeMoney();
}

public interface Company {
    String unit();
}

public class Corp implements Organization, Business, Company {

    private DevDep devDep = new DevDep();
    private SaleDep saleDep = new SaleDep();

    public String develop() {
        return devDep.work();
    }

    public String sale() {
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
        return saleDep.work();
    }

    public String makeMoney() {
        return "Making a lot of money.";
    }

    public String unit() {
        return "We are a team.";
    }

    public String goal() {
        return "Do something good.";
    }

    class DevDep implements Department {
        public String work() {
            return "开发部正在工作...";
        }
    }

    class SaleDep implements Department {
        public String work() {
            return "销售部正在工作...";
        }
    }
}
```

【习题 9-8】定义一个接口 Standard，在接口中定义几个属性。

【分析】此题主要考查接口定义。

【核心代码】

```
public interface Standard {
    String name = "standard";
    String level = "particular";
    String authorize = "official sanction";
    Boolean feasible = true;
}
```

【习题 9-9】定义一个打印机类 Printer，在其主方法中调用一个内部类 PrintHead，负责打印文字。

【分析】此题主要考查静态内部类的实现。

【核心代码】

```
public class Printer {
    public static void main(String[] args) {
        Scanner reader = new Scanner(System.in);
        System.out.println("请输入打印内容：");
        String content = reader.next();

        PrintHead ph = new PrintHead();
        ph.strike(content);
    }

    static class PrintHead {
        public void strike(String s) {
            System.out.println("Striking words");
            System.out.println(s);
        }
    }
}
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

【习题 9-10】在第 9 题的基础上，编写一个彩色打印机，实现一个彩色打印头继承 PrintHead。

【分析】此题主要考查内部类的继承。

【核心代码】

```
public class ColorPrinter {
    public static void main(String[] args) {
        Scanner reader = new Scanner(System.in);
        System.out.println("请输入打印内容：");
        String content = reader.next();
        boolean isColor = reader.nextBoolean();
        PrintHead ph;
        if (isColor) {
            ph = new ColorPrintHead();
            ph.strike(content);
        } else {
            ph = new PrintHead();
            ph.strike(content);
        }

    }

    static class PrintHead {
        public void strike(String s) {
            System.out.println("Striking words");
            System.out.println(s);
        }
    }
    static class ColorPrintHead extends PrintHead {
        public void strike(String s) {
            System.out.println("Striking color words");
            System.out.println(s);
        }
    }
}
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第 2 篇　高级编程篇

第 10 章　图形界面开发基础

图形用户界面（Graphical User Interface）简称 GUI，它提供了一系列更加直观、友好地与用户进行互动的方式。Java 中对图形用户界面进行操作的包主要有 AWT 和 Swing 两种。其中，AWT 是 Java 图形用户界面的基本包，而 Swing 是对于 AWT 功能上的一些扩展。本章主要介绍图形用户界面的开发基础。



10.1　AWT 简介

AWT 是抽象窗口工具箱，其中包括了图形界面开发的基本类库。如果将图形界面开发作为一幅图画的创作的话，那么 AWT 就相当于画画所使用到的水彩笔工具包，在这里面有多种能够使用到，或者可以进行搭配的颜色笔。

AWT 主要包括容器、布局管理器、事件处理、基本组件四部分，它们各自有不同的作用。

- 容器：用于放置基本组件，类似于画纸或者是画板。
- 布局管理器：对容器中所放置的组件进行布局管理，也就是什么地方应该画什么东西，构思巧妙，才能画出美丽的图画。
- 事件处理：采用事件处理机制，可以使得用户能通过图形界面和系统进行很好的交互。
- 基本组件：图形界面组成的基本元素，也就是将要画到画板上的物体。



10.2　容器

容器是用来放置组件的一种特殊组件，其功能相当于一个鱼缸，其中放置的水、水草、各种各样的鱼就相当于组件，将各种组件按照不同的顺序或者是样式摆放到容器中。所有的容器类都是 Container 的子类，两个常用类是 Frame 和 Panel。本节就主要介绍这两个不同的容器类。

10.2.1　窗口

Frame 是带有边框的窗口，相当于一个窗子，各边都有框，玻璃就镶在边框上。Frame 可以不依赖于任何的组件而独立存在。用户可以设置窗口的大小及标题等。在该类中，也为窗口的操作定义了不同的方法，常用方法如下。

- boolean isVisible()：该方法的作用是测试窗口是否可见。
- boolean isShowing()：该方法的作用是检查组件是否显示在屏幕上，如果该组件正显示在屏幕上，则返回 true。
- boolean isUndecorated()：该方法的作用是如果窗口没有修饰的话返回 true。
- boolean isEnable()：该方法的作用是检查组件是否被激活，当被激活的时候才能处理各种事件。
- void toFront()：该方法的作用是将该窗口显示到其他任何窗口的前面。
- void toBack()：该方法与 toFront()方法相反，如果窗口是可视的，则将此窗口置于后方，

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

如果它是焦点窗口或活动窗口，则会导致丢失焦点或活动状态。

- Point getLocation()：返回该组件左上角的坐标值，将该坐标值封装在一个 Point 对象中。
- Point getLocationOnScreen()：该方法的作用是返回该组件左上角的坐标值，使用屏幕坐标。
- Dimension getSize()：该方法的作用是获得组件的大小。
- void setBounds(int x,int y,int width,int height)：该方法的作用是移动组件并设置组件的大小，左上角由 x 和 y 指定坐标，而 width 和 height 指定组件的宽度和高度。
- void setVisible(boolean b)：该方法的作用是根据 b 的取值来显示或隐藏窗体。
- void setLocation(int x,int y)：该方法的作用是重定位组件的坐标。
- void setLocation(Point p)：该方法的作用是把组件重定位，组件的位置由 Point 指定。
- void setSize(int width,int height)：该方法的作用是重新设置组件的大小。
- void setSize(Dimension d)：该方法的作用是重设组件大小。
- void setResizable(boolean b)：该方法的作用是决定用户是否可以缩放组件。
- void setTitle(String s)：该方法的作用是将窗体的标题设为 s。
- void setIconImage(Image image)：该方法的作用是把图像对象用做框架图标。
- void setUndecorated(boolean b)：该方法的作用是清除对窗口的修饰。
- void setEnable(boolean b)：该方法的作用是根据 b 的值来激活或禁用一个组件。
- int getExtendedState()：该方法的作用是获得窗口状态。
- void setExtendedState(int state)：该方法的作用是设置窗口状态。

上面介绍了一些 Frame 类的常用方法，下面就使用其中的一些方法编写一个 Frame 窗口的范例。

【范例 10-1】示例代码 10-1 是一个测试 Frame 窗口的程序。

示例代码 10-1

```
01  import java.awt.Dimension;                          //加载所能使用到的类
02  import java.awt.Frame;
03  import java.awt.Graphics;
04  import java.awt.Toolkit;
05  import java.awt.event.WindowAdapter;
06
07  public class FrameTest {
08      public static void main(String[] args) {
09          Frame frame=new NewFrame();                 //创建一个窗体对象
10          frame.setVisible(true);                     //显示窗体
11      }
12  }
13  class NewFrame extends Frame                        //继承自 JFrame 类
14  {
15      NewFrame()                                      //构造函数
16      {
17          setTitle("这是一个在屏幕的中间位置显示的窗口");   //设置窗体标题
18          Toolkit tk=Toolkit.getDefaultToolkit();     //取得工具类
19          Dimension screenSize=tk.getScreenSize();    //取得显示器尺寸
20          int screenHeight=screenSize.height;
21          int screenWidth=screenSize.width;
22          setSize(screenWidth/2,screenHeight/2);      //设置窗体大小
23          setLocation(screenWidth/4,screenHeight/4);  //设置窗体位置
24          this.repaint();                             //重绘窗体
25      }
26  }
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

【运行结果】程序运行结果如图 10-1 所示。

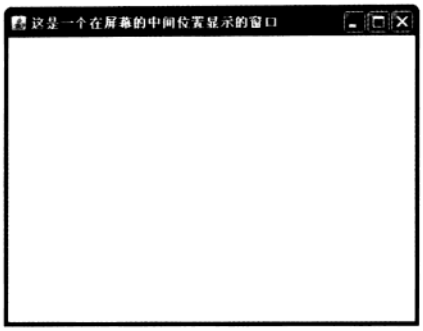


图 10-1　创建一个在屏幕中间显示的窗口

【代码解析】以上程序中，创建一个 NewFrame 类，并继承 Frame。然后创建该类的构造方法，并在构造方法中设置窗口的显示标题，设置窗口的大小及显示位置。最后在测试类 FrameTest 中调用并显示。



注意：Fram 窗口在创建的时候是不可见的，此时需要调用 setVisible()方法将其设置为可见状态。

10.2.2　面板

面板是 Panel 类，Panel 没有任何边框，而且需要存放在其他的容器中才能使用。就像是一张画纸一样，没有任何的边框，而且需要将它固定在画板上或者其他物体上才能使用。该容器只是包含在其他容器中的一个区域中，一般使用来将多个组件组合在一个区域中。面板本身可以存放其他的组件，还可以在面板上完成绘制工作。

【范例 10-2】示例代码 10-2 是一个测试 Panel 面板的程序。

示例代码 10-2

```
01  public class PanelTest
02  {
03      public static void main(String[] args)          //主方法
04      {
05          Frame f=new Frame("新建 Panel");
06          Panel p1=new Panel();
07          p1.setBackground(Color.red);
08
09          Panel p2=new Panel();
10          p1.setBackground(Color.blue);
11
12          Panel p3=new Panel();
13          p1.setBackground(Color.yellow);
14
15          f.add(p1,BorderLayout.NORTH);
16          f.add(p2,BorderLayout.CENTER);
17          f.add(p3,BorderLayout.SOUTH);
18
20          f.setSize(200,200);
21          f.setVisible(true);
22      }
23
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
24  }
```

【运行结果】程序运行结果如图 10-2 所示。

【代码解析】在以上程序中，首先创建窗口 Frame，然后分别创建三个不同的面板容器，分别设置不同的颜色，最后添加到窗口 Frame 中并显示。

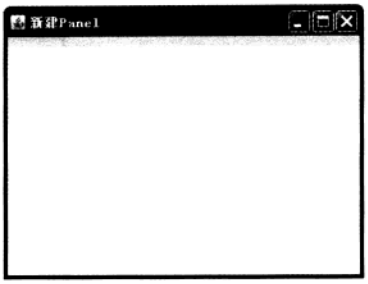


图 10-2　Panel 示例运行效果

10.3　组件

构成图形用户界面的基本元素就是组件。

10.3.1　按钮和标签

按钮（Button）和标签（Label）是图形界面中比较常用的组件，用户可以根据标签的提示单击按钮来触发一系列的动作。其中最常见的就是登录框，在输入框左边的就是标签，填写完成后，可以单击下面的“登录”按钮进行登录。

按钮中一些比较常用的方法如下。

- void addActionListener(ActionListener l)：该方法的作用是添加指定的动作侦听器，以接收发自此按钮的动作事件。
- void addNotify() ：创建按钮的同位体。
- AccessibleContext getAccessibleContext() ：该方法的作用是获取与此 Button 相关的 AccessibleContext。
- String getActionCommand()：该方法的作用是返回此按钮激发的动作事件的命令名称。
- ActionListener[] getActionListeners() ：该方法的作用是返回在此按钮上注册的所有动作侦听器的一个数组。
- String getLabel()：该方法的作用是获取此按钮的标签。
- protected String paramString()：该方法的作用是返回此 Button 状态的字符串表示形式。
- protected void processActionEvent(ActionEvent e) ：该方法的作用是处理发生在此按钮上的动作事件，方法是将这些事件指派给所有已注册的 ActionListener 对象。
- protected void processEvent(AWTEvent e) ：该方法的作用是处理此按钮上的事件。
- void removeActionListener(ActionListener l)：该方法的作用是移除指定的动作侦听器，以便它不再接收发自此按钮的动作事件。
- void setActionCommand(String command)：该方法的作用是设置此按钮激发的动作事件的命令名称。
- void setLabel(String label) ：该方法的作用是将按钮的标签设置为指定的字符串。

【范例 10-3】示例代码 10-3 是一个测试 Button 的程序。

示例代码 10-3

```
01  public class NewButton {
02      public static void main(String[] args) {
03          Frame f=new Frame("Button");
04          Button b=new Button();
05          b.setLabel("这是一个按钮，功能是关闭窗口");
06
07          b.addActionListener(new ActionListener(){
08
09              public void actionPerformed(ActionEvent e) {
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
                    // TODO Auto-generated method stub
                    System.exit(1);
                }
            });
            f.add(b);
            f.setSize(200,200);
            f.setVisible(true);
        }
    }
```

【运行结果】程序运行结果如图 10-3 所示。

【代码解析】以上程序中，首先创建了一个窗口容器 Frame，在该容器中创建一个按钮，并设置按钮的显示文本及大小等，然后为按钮设置一个事件，也就是在单击该按钮时所发生的动作是关闭窗口，最后设置窗口可见。

Label 标签也有一些常用的方法，如下所示。

- void addNotify()：创建此标签的同位体。
- AccessibleContext getAccessibleContext()：该方法的作用是获取与此标签相关的 AccessibleContext。
- int getAlignment()：该方法的作用是获取此标签的当前对齐方式。
- String getText()：该方法的作用是获取此标签的文本。
- protected String paramString()：该方法的作用是返回一个表示此 Label 状态的字符串。
- void setAlignment(int alignment)：该方法的作用是将此标签的对齐方式设置为指定的方式。
- void setText(String text)：该方法的作用是将此标签的文本设置为指定的文本。

【范例 10-4】示例代码 10-4 是一个测试 Label 的程序。

示例代码 10-4

```
public class NewJLabel {
    public static void main(String[] args) {
        Frame f = new Frame("这是一个标签示例");

        Label label = new Label("我是一个标签"); //创建一个 JLabel 标签
        label.setAlignment(Label.CENTER);

        f.add(label);
        f.setSize(200,200);
        f.setVisible(true);

    }
}
```

【运行结果】程序运行结果如图 10-4 所示。



图 10-3　Button 效果



图 10-4　Label 效果

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

【代码解析】以上程序中，首先创建了一个窗口容器 Frame，在该窗口中创建标签，并设置标签的显示内容。将标签添加到窗口中，最后设置窗口的大小及可见。

10.3.2 复选框

复选框是 Checkbox，属于一种选择按钮，可以供用户进行相关选项的选择。如果要实现多选一，需要结合 CheckboxGroup 类使用；如果要实现的是多选按钮，就不需要给多个 Chekbox 设置分组。

Checkbox 的一些常用方法如下。

- void addComponentListener(ComponentListener l)：添加指定的组件侦听器，以接收发自此组件的组件事件。
- void addNotify()：创建 Checkbox 的同位体。该同位体允许更改 Checkbox 的外观，而不更改其功能。
- AccessibleContext getAccessibleContext()：获取与此组件相关的 AccessibleContext。
- CheckboxGroup getCheckboxGroup()：确定该复选框的组。
- ItemListener[] GetItemListeners()：返回已在此复选框上注册的所有项监听器所组成的数组。
- String getLabel()：获得此复选框的标签。
- Object[] getSelectedObjects()：返回包含复选框标签的数组，如果没有选定复选框的话，则返回 null。
- boolean getState()：确定此复选框是处于“开”的状态还是“关”的状态。
- protected String paramString()：返回表示此 Checkbox 状态的关键字。
- protected void processEvent(AWTEvent e)：处理关于该复选框的事件。
- void setCheckboxGroup(CheckboxGroup g)：将该复选框的组设置为指定复选框。

【范例 10-5】示例代码 10-5 是一个测试 Checkbox 的程序。

示例代码 10-5

```
01  public class NewCheckbox {
02      public static void main(String[] args) {
03          Frame f=new Frame("Checkbox 示例");
04          f.setLayout(new GridLayout(2,3));
05          Label lb=new Label("个人爱好");
06
07          Checkbox c1=new Checkbox("读书");
08          Checkbox c2=new Checkbox("打球");
09          Checkbox c3=new Checkbox("上网");
10          Checkbox c4=new Checkbox("看电视");
11          Checkbox c5=new Checkbox("游泳");
12
13          f.add(lb);
14          f.add(c1);
15          f.add(c2);
16          f.add(c3);
17          f.add(c4);
18          f.add(c5);
19
20          f.setSize(200,100);
21          f.setVisible(true);
22
23
24      }
25  }
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



【运行结果】程序运行结果如图 10-5 所示。

【代码解析】以上程序中，首先创创建一个容器窗口 Frame，接下来创建五个复选框，并将五个复选框分别加入到 Frame 中，最后设置 Frame 的大小及可见属性。

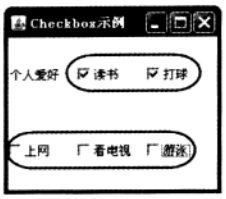


图 10-5　Checkbox 示例效果

10.3.3　文本域

文本域（TextField）顾名思义就是用来书写文字的组件。文本框中提供了文本的编辑功能，TextField 的一些常用方法如下。

- void addActionListener(ActionListener l)：该方法的作用是添加指定的操作侦听器，以从此文本字段接收操作事件。
- void addNotify()：该方法的作用是创建 TextField 的同位体。
- boolean echoCharIsSet()：该方法的作用是指示此文本字段是否有一个回显字符集。
- AccessibleContext getAccessibleContext()：该方法的作用是获取与此 TextField 关联的 AccessibleContext。
- ActionListener[] getActionListeners()：该方法的作用是返回此文本字段上已注册的所有操作侦听器的数组。
- int getColumns()：该方法的作用是获取此文本字段中的列数。
- char getEchoChar()：该方法的作用是获取用于回显的字符。
- <T extends EventListener> T[]getListeners(Class<T> listenerType) ：该方法的作用是返回当前已在此 TextField 上注册的所有监听器对象的数组。
- Dimension getMinimumSize()：该方法的作用是获取此文本字段的最小尺寸。
- Dimension getMinimumSize(int columns)：该方法的作用是获取具有指定列数的文本字段的最小尺寸。
- Dimension getPreferredSize()：该方法的作用是获取此文本字段的首选大小。
- Dimension getPreferredSize(int columns)：该方法的作用是获取具有指定列数的文本字段的首选大小。
- protected String paramString()：返回表示此 TextField 状态的字符串，此方法仅用于调试。
- protected void processActionEvent(ActionEvent e)：该方法的作用是处理发生在此文本字段上的操作事件：将这些事件指派给所有已注册的 ActionListener 对象。
- protected void processEvent(AWTEvent e)：该方法的作用是处理此文本字段上的事件。
- void removeActionListener(ActionListener l)：该方法的作用是移除指定的操作侦听器，不再从此文本字段接收操作事件。
- void setColumns(int columns) ：该方法的作用是设置此文本字段中的列数。
- void setEchoChar(char c)：该方法的作用是设置此文本字段的回显字符。
- void setText(String t)：该方法的作用是将此文本组件显示的文本设置为指定文本。

【范例 10-6】示例代码 10-6 是一个测试 Checkbox 的程序。

示例代码 10-6

```
01  public class NewTextFiled {
02      public static void main(String[] args) {
03          Frame f=new Frame("这是一个文本域示例");
04          f.setLayout(new GridLayout(2,3));
05          Label lb=new Label("请输入文本");
06          lb.setAlignment(Label.RIGHT);
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
07            TextField tf=new TextField();
08            tf.setBackground(Color.gray);
09
10            f.add(lb);
11            f.add(tf);
12            f.setSize(200,60);
13            f.setVisible(true);
14        }
15    }
```

【运行结果】程序运行结果如图 10-6 所示。



图 10-6　TextFiled 效果

【代码解析】以上程序中，首先创建一个容器窗口 Frame，接下来创建一个文本域，然后将该文本域添加到 Frame 中，最后设置 Frame 的大小及可见属性。

以上的文本域输入只能为一行，有字数的限制。在实际应用中，这种单行输入的文本域也就只会出现在注册或者是登录等窗口中。如果需要输入非常多的文字，就需要使用到多行文本域（TextArea）。多行文本域的常用的方法如下。

- void addNotify() ：该方法的作用是创建 TextArea 的同位体。
- void append(String str)：该方法的作用是将给定文本追加到文本域的当前文本。
- AccessibleContext getAccessibleContext()：该方法的作用是返回与此 TextArea 相关的 AccessibleContext。
- int getColumns()：该方法的作用是返回此文本域中的列数。
- Dimension getMinimumSize()：该方法的作用是确定此文本域的最小大小。
- Dimension getMinimumSize(int rows, int columns)：该方法的作用是确定具有指定行数和列数的文本域的最小大小。
- Dimension getPreferredSize() ：该方法的作用是确定此文本域的首选大小。
- Dimension getPreferredSize(int rows, int columns)：该方法的作用是确定具有指定行数和列数的文本域的首选大小。
- int getRows()：该方法的作用是返回此文本域的行数。
- int getScrollbarVisibility()：该方法的作用是返回指示文本域使用何种滚动条的枚举值。
- void insert(String str, int pos)：该方法的作用是在此文本域的指定位置插入指定文本。
- void replaceRange(String str, int start, int end)：该方法的作用是用指定替换文本替换指定开始位置与结束位置之间的文本。
- void setColumns(int columns)：该方法的作用是设置此文本域的列数。
- void setRows(int rows)：该方法的作用是设置此文本域的行数。

【范例 10-7】示例代码 10-7 是一个测试 TextArea 的程序。

示例代码 10-7

```
01  public class NewTextArea {
02      public static void main(String[] args) {
03
04          Frame f=new Frame("这是一个多行文本域示例");
05          f.setLayout(new GridLayout(2,3));
06          Label lb=new Label("请输入文本，可以换行");
07          lb.setAlignment(Label.RIGHT);
08          TextArea tf=new TextArea();
09          tf.setBackground(Color.gray);
10
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
11            f.add(lb);
12            f.add(tf);
13            f.setSize(200,60);
14            f.setVisible(true);
15        }
16    }
```

【运行结果】程序运行结果如图 10-7 所示。

【代码解析】以上程序中，首先创建一个容器窗口 Frame，接下来创建一个多行文本域（TextArea），然后将该文本域添加到 Frame 中，最后设置 Frame 的大小及可见属性。

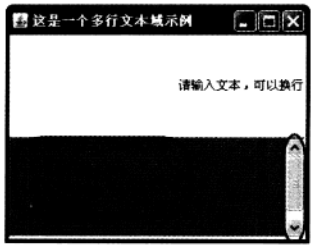


图 10-7　多行文本域效果

10.3.4　列表

列表 List 组件可以列举多个可供选择的文本选项，其中包括有单选和多选两种方式。所列举出来的选项是分为多行显示的，并且只允许选择，不允许对选项进行修改。List 类的一些常用方法如下所示。

- void add(String item) ：向滚动列表的末尾添加指定的项。
- void add(String item, int index) ：向滚动列表中索引指示的位置添加指定的项。
- void addActionListener(ActionListener l) ：添加指定的动作侦听器，以从此列表接收动作事件。
- void addItemListener(ItemListener l) ：添加指定的项侦听器，以接收此列表的项事件。
- void addNotify()：创建列表的同位体。
- void deselect(int index) ：取消选择指定索引处的项。
- AccessibleContext getAccessibleContext()：获取与此 List 关联的 AccessibleContext。
- ActionListener[] getActionListeners()：返回已在此列表上注册的所有动作侦听器的数组。
- String getItem(int index) ：获取与指定索引关联的项。
- int getItemCount()：获取列表中的项数。
- ItemListener[] getItemListeners()：返回已在此列表上注册的所有项侦听器的数组。
- String[] getItems()：获取列表中的项。
- <T extends EventListener> T[] getListeners(Class<T> listenerType) ：返回目前已在此 List 上注册为 FooListener 的所有对象的数组。
- Dimension getMinimumSize()：确定此滚动列表的最小大小。
- Dimension getMinimumSize(int rows) ：获取具有指定行数的列表的最少维数。
- Dimension getPreferredSize()：获取此滚动列表的首选大小。
- Dimension getPreferredSize(int rows) ：获取具有指定行数的列表的首选大小。
- int getRows()：获取此列表中的可视行数。
- int getSelectedIndex()：获取列表中选中项的索引。
- int[] getSelectedIndexes()：获取列表中选中的索引。
- String getSelectedItem()：获取此滚动列表中选中的项。
- String[] getSelectedItems()：获取此滚动列表中选中项的数组。
- Object[] getSelectedObjects()：获取对象数组中此滚动列表的选中项。
- int getVisibleIndex()：获取上次由 makeVisible 方法使其可视的项的索引。
- boolean isIndexSelected(int index) ：确定是否已选中此滚动列表中的指定项。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

- boolean isMultipleMode()：确定此列表是否允许进行多项选择。
- boolean isSelected(int index)：已过时。从 JDK version 1.1 开始，由 isIndexSelected(int) 取代。
- void makeVisible(int index) ：使指定索引处的项可视。
- protected void processActionEvent(ActionEvent e) ：处理发生在此列表上的动作事件，方法是将这些事件指派给所有已注册的 ActionListener 对象。
- protected void processEvent(AWTEvent e) ：此滚动列表的进程事件。
- protected void processItemEvent(ItemEvent e) ：处理发生在此列表上的项事件，方法是将这些事件指派给所有已注册的 ItemListener 对象。
- void remove(int position) ：从此滚动列表中移除指定位置处的项。
- void remove(String item) ：从列表中移除项的第一次出现。
- void removeActionListener(ActionListener l) ：移除指定的动作侦听器，以便不再从此列表接收动作事件。
- void removeAll()：从此列表中移除所有项。
- void removeItemListener(ItemListener listener) ：移除指定的项侦听器，以便不再从此列表接收项事件。
- void removeNotify()：移除此列表的同位体。
- void replaceItem(String newValue, int index) ：使用新字符串替换滚动列表中指定索引处的项。
- void select(int index) ：选择滚动列表中指定索引处的项。
- void setMultipleMode(boolean b) ：设置确定此列表是否允许进行多项选择的标志。

【范例 10-8】示例代码 10-8 是一个测试 List 的程序。

示例代码 10-8

```
01  public class NewList {
02      public static void main(String[] args) {
03          //创建窗口容器并设置
04          Frame f=new Frame("这是一个列表示例");
05          f.setLayout(new GridLayout(1,2));
06          Label lb=new Label("请选择右边的年份");
07          lb.setAlignment(Label.RIGHT);
08          //创建列表
09          List list=new List(7,true);
10          list.add("2007 年");
11          list.add("2008 年");
12          list.add("2009 年");
13          list.add("2010 年");
14          //将列表添加到窗口中
15          f.add(lb);
16          f.add(list);
17          //设置窗口可见
18          f.setSize(200,100);
19          f.setVisible(true);
20      }
21  }
```



图 10-8 列表效果

【运行结果】程序运行结果如图 10-8 所示。

【代码解析】以上程序中，首先创建一个容器窗口 Frame，接下来创建一个列表，并设置列表为可多选，然后将该列表添加到 Frame 中，最后设置 Frame 的大小及可见属性。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



10.4　布局管理器

布局管理器用来对容器中的组件进行布局，相当于装修房屋，什么地方该放沙发、什么地方放茶几，还有要放置多大的沙发、茶几等。布局管理器可以设置组件的大小和位置。每一个容器在创建的时候都会有一个默认的布局管理器。

10.4.1　网格布局管理器

网格布局管理器 GridLayout，好比是已经设置好了由一定的行和列来划分的网格，只需要向这些网格中放置组件就可以了。在容器中，使用网格布局管理器，需要按照从上到下、从左到右的顺序依次向网格中添加组件。GridLayout 中提供的一些方法如下所示。

- void addLayoutComponent(String name, Component comp)：该方法的作用是将具有指定名称的指定组件添加到布局。
- int getColumns()：该方法的作用是获取此布局中的列数。
- int getHgap()：该方法的作用是获取组件之间的水平间距。
- int getRows()：该方法的作用是获取此布局中的行数。
- int getVgap()：该方法的作用是获取组件之间的垂直间距。
- void layoutContainer(Container parent)：该方法的作用是使用此布局布置指定容器。
- Dimension minimumLayoutSize(Container parent)：该方法的作用是使用此网格布局确定最小大小的容器参数。
- Dimension preferredLayoutSize(Container parent)：该方法的作用是使用此网格布局确定容器参数的首选大小。
- void removeLayoutComponent(Component comp)：该方法的作用是从布局移除指定组件。
- void setColumns(int cols)：该方法的作用是将此布局中的列数设置为指定值。
- void setHgap(int hgap)：该方法的作用是将组件之间的水平间距设置为指定值。
- void setRows(int rows)：该方法的作用是将此布局中的行数设置为指定值。
- void setVgap(int vgap)：该方法的作用是将组件之间的垂直间距设置为指定值。
- String toString()：返回此网格布局的值的字符串表示形式。

【范例 10-9】示例代码 10-9 是一个测试网格布局管理器的示例。

示例代码 10-9

```
01  public class NewGridLayout {
02      public static void main(String[] args) {
03          //创建窗口容器并设置
04          Frame f=new Frame("这是一个网格布局示例");
05          f.setLayout(new GridLayout(3,3));
06          //使用 for 循环为窗口添加按钮
07          for(int i=0;i<9;i++){
08              f.add(new Button("按钮"+(i+1)));
09
10          }
11          //设置窗口的大小，以及可见
12          f.setSize(200,200);
13          f.setVisible(true);
14
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
15        }
16    }
```

【运行结果】程序运行结果如图 10-9 所示。



图 10-9　网格布局管理器示例效果

【代码解析】以上程序中，首先创建一个容器窗口 Frame，然后设置窗口的布局格式为网格布局管理器，接下来使用 for 循环创建九个按钮，并依次将其添加到窗口中，最后设置 Frame 的大小及可见属性。

如果需要设置组件之间的间距或者是设置每个组件不一定只占用一个格子，就需要使用 GridBagLayout 布局管理器，该管理器是在网格布局管理器上的扩展，其中提供了更加复杂的布局管理，例如，一个组件可以跨越多个行列放置在网格内，等等。而且，其中还引入了 GridBagConstraints 类，一些比较常用的属性如下所示。

- int anchor：指定组件在区域中所存放的位置。
- int fill：指定组件的存放方式。
- int Gridheight：指定组件的高度，单位是网格个数。
- Int gridWidth：指定组件的宽度，单位是网格的个数。
- int gridx：指定组件的横向坐标，单位是网格的个数。
- int ipadx：指定组件间的横向间距。
- Int ipady：指定组件间的纵向间距。
- GridBagLayout 的一些比较常用的方法如下所示。
- void addLayoutComponent(Component comp, Object constraints)：该方法的作用是使用指定 constraints 对象将指定组件添加到布局中。
- protected void adjustForGravity(GridBagConstraints constraints, Rectangle r)：该方法的作用是根据约束几何结构和填充，将 x、y、宽度和高度四个字段调整为正确值。
- protected void arrangeGrid(Container parent)：该方法的作用是布置网格。
- GridBagConstraints getConstraints(Component comp)：该方法的作用是获取指定组件的约束。
- floatgetLayoutAlignmentX(Container parent)：该方法的作用是返回沿 X 轴的对齐方式。
- float getLayoutAlignmentY(Container parent)：该方法的作用是返回沿 Y 轴的对齐方式。
- int[][] getLayoutDimensions()：该方法的作用是确定布局网格的列宽度和行高度。
- protected GridBagLayoutInfo getLayoutInfo(Container parent, int sizeflag)：该方法作用是为当前受管子布局器的集合填充 GridBagLayoutInfo 的实例。
- Point getLayoutOrigin()：该方法的作用是在目标容器的图形坐标空间确定布局区域的原点。
- double[][] getLayoutWeights()：该方法的作用是确定布局网格的行与列的权重。
- protected Dimension getMinSize(Container parent, GridBagLayoutInfo info)：该方法的作用是基于 getLayoutInfo 中的信息计算其所有者的最小大小。
- void invalidateLayout(Container target)：该方法的作用是使布局失效，指示如果布局管理器缓存了信息，则应该将其丢弃。
- void layoutContainer(Container parent)：该方法的作用是使用此网格包布局布置指定容器。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



- Point location(int x, int y)：确定在布局网格中哪个单元包含由 (x, y) 指定的点。
- protected GridBagConstraints lookupConstraints(Component comp)：检索指定组件的约束。
- Dimension maximumLayoutSize(Container target)：该方法的作用是在给出指定目标容器中的组件的前提下，返回此布局的最大维数。
- Dimension minimumLayoutSize(Container parent)：该方法的作用是使用此网格包布局确定 parent 容器的最小大小。
- Dimension preferredLayoutSize(Container parent)：该方法的作用是使用此网格包布局确定 parent 容器的首选大小。
- void removeLayoutComponent(Component comp)：该方法的作用是从此布局移除指定组件。
- void setConstraints(Component comp, GridBagConstraints constraints)：该方法的作用是设置此布局中指定组件的约束条件。
- String toString()：该方法的作用是返回此网格包布局的值的字符串表示形式。

【范例 10-10】示例代码 10-10 是一个测试网格布局管理器的示例。

示例代码 10-10

```
01  public class NewGridBagLayout {
02
03          public static void main(String[] args) {
04          //创建窗口容器并设置布局管理器
05          Frame f=new Frame("这是GridBagLayout示例");
06          f.setLayout(new GridBagLayout());
07          GridBagConstraints gbcs=new GridBagConstraints();
08          //设置横向坐标和纵向坐标，并加入按钮
09          gbcs.gridx=0;
10          gbcs.gridy=0;
11          f.add(new Button("按钮一"),gbcs);
12          //设置横向坐标和纵向坐标，并加入按钮
13          gbcs.gridx=1;
14          gbcs.gridy=0;
15          f.add(new Button("按钮二"),gbcs);
16          //设置横向坐标和纵向坐标，并加入按钮
17          gbcs.gridx=0;
18          gbcs.gridy=1;
19          f.add(new Button("按钮三"),gbcs);
20          //设置横向坐标和纵向坐标，并加入按钮
21          gbcs.gridx=1;
22          gbcs.gridy=1;
23          f.add(new Button("按钮四"),gbcs);
24
25          f.setSize(200,200);
26          f.setVisible(true);
27
28      }
29  }
```

【运行结果】程序运行结果如图 10-10 所示。

【代码解析】以上程序中，首先创建一个容器窗口 Frame，然后设置窗口的布局格式为 GridBagLayout，接下来创建 GridBagConstraints 的对象，并依次为其添加按钮，使用 gridx 和 gridy 方法指定组件的横向和纵向坐标位置，最后设置 Frame 的大小及可见属性。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



图 10-10　GridBagLayout 布局管理器效果

10.4.2　顺序布局管理器

顺序布局管理器（FlowLayout）是默认的布局管理器，该布局管理器可以将组件按照从左到右、从上到下的顺序来排列，并且顺序布局管理器会在默认的情况下，尽量将组件居中放置。如果一行放满之后，就会自动将其他的组件放到下一行中显示。顺序布局管理器中提供了一些常用的方法，如下所示。

- void addLayoutComponent(String name, Component comp)：该方法的作用是将指定的组件添加到布局中。
- int getAlignment()：该方法的作用是获取此布局的对齐方式。
- boolean getAlignOnBaseline()：该方法的作用是如果组件将沿其基线垂直对齐，则返回 true。
- int getHgap()：该方法的作用是获取组件之间，以及组件与 Container 的边之间的水平间隙。
- int getVgap()：该方法的作用是获取组件之间，以及组件与 Container 的边之间的垂直间隙。
- void layoutContainer(Container target)：该方法的作用是布置该容器。
- Dimension minimumLayoutSize(Container target)：该方法的作用是返回需要布置可视组件的最小尺寸，该组件包含在指定的目标容器中。
- Dimension preferredLayoutSize(Container target)：该方法的作用是返回给定容器中可视组件的首选尺寸。
- void removeLayoutComponent(Component comp)：该方法的作用是从布局中移除指定的组件。
- void setAlignment(int align)：该方法的作用是设置此布局的对齐方式。
- void setAlignOnBaseline(boolean alignOnBaseline)：该方法的作用是设置组件是否应该沿着其基线垂直对齐。
- void setHgap(int hgap)：该方法的作用是设置组件之间，以及组件与 Container 的边之间的水平间隙。
- void setVgap(int vgap)：该方法的作用是设置组件之间，以及组件与 Container 的边之间的垂直间隙。
- String toString()：该方法的作用是返回此 FlowLayout 对象及其值的字符串表示形式。

【范例 10-11】示例代码 10-11 是一个测试顺序布局管理器的示例。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



示例代码 10-11

```
01  public class NewFlowLayout extends Frame {
02      //定义两个按钮组件
03      Button b1=new Button("按钮一");
04      Button b2=new Button("按钮二");
05      Button b3=new Button("按钮三");
06      Button b4=new Button("按钮四");
07
08      NewFlowLayout()
09      {
10          //设置窗口名称
11          this.setTitle("FlowLayout 布局管理器");
12          //设置布局管理器为 FlowLayout
13          this.setLayout(new FlowLayout());
14          //添加按钮
15          this.add(b1);
16          this.add(b2);
17          this.add(b3);
18          this.add(b4);
19          //设置窗口大小及显示位置
20          this.setBounds(100,100,250,250);
21          //设置窗口可见
22          this.setVisible(true);
23      }
24      public static void main(String[] args) {
25          new NewFlowLayout();
26      }
27
28  }
```

【运行结果】程序运行结果如图 10-11 所示。



图 10-11　FlowLayout 示例效果

【代码解析】以上程序中，创建了一个类 NewFlowLayout 并继承 Frame。创建了四个按钮，并将按钮添加到窗口容器中，设置窗口的大小及显示位置，设置窗口可显示，最后在主方法中调用。

在上面的示例中，仅仅只有四个按钮，所以不容易看出布局管理器的样式，可使用事件监听的方式，为窗口依次添加几个按钮，在单击“添加”按钮时，会自动增加按钮的个数，这样一来，就会清楚地显示流式布局管理器的样式。

【范例 10-12】示例代码 10-12 是一个改进测试顺序布局管理器的示例。

示例代码 10-12

```
01  public class FlowLayoutTest extends Frame implements ActionListener{
02      Button b=new Button("添加");
03      int i;
04      FlowLayoutTest()
05      {
06          //设置窗口名称
07          this.setTitle("FlowLayout 布局管理器");
08          //设置布局管理器为 FlowLayout
09          this.setLayout(new FlowLayout());
10          //添加按钮
11          this.add(b);
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
12            b.addActionListener(this);
13            //设置窗口大小
14            this.setBounds(100,100,250,250);
15            //设置窗口可见
16            this.setVisible(true);
17        }
18        public static void main(String[] args) {
19            new FlowLayoutTest();
20        }
21        public void actionPerformed(ActionEvent e) {
22
23            i++;
24            Button bi=new Button("按钮"+i);
25            this.add(bi);
26            this.show(true);
27
28        }
29
30    }
```



图 10-12　FlowLayoutTest 运行效果

【运行结果】程序运行结果如图 10-12 所示。

【代码解析】以上程序中，首先创建类 FlowLayoutTest，继承 Frame 类，并创建一个按钮，接下来设置窗口的大小及显示位置，添加按钮并为按钮添加监听。设置监听为每单击一次该按钮，生成一个按钮，同时刷新窗口。最后在主方法中进行调用显示窗口。

从以上的程序中可以清楚地看出，FlowLayout 布局管理器是将组件以从左到右、从上到下的顺序依次进行排列的，而且尽可能地将组件居中放置。

10.4.3　边界布局管理器

边界布局管理器（BorderLayout）将一个容器分为五个部分，而且只允许在容器中放置五个组件。该布局管理器类似于麻将桌，打麻将的时候，桌子就是被分为五个部分的，四个人各占一面，中间则是打出的牌。边界布局管理器将容器分为东、西、南、北和中五部分，使用 EAST、WEST、SOUTH、NORTH、CENTER 五个常量来表示。BorderLayout 中提供了一些常用的方法，如下所示。

- void addLayoutComponent(Component comp, Object constraints)：使用指定的约束对象将指定组件添加到布局中。
- Object getConstraints(Component comp) ：获取指定组件的约束。
- int getHgap()：返回组件之间的水平间距。
- float getLayoutAlignmentX(Container parent)：返回沿 X 轴的对齐方式。
- float getLayoutAlignmentY(Container parent)：返回沿 Y 轴的对齐方式。
- Component getLayoutComponent(Container target, Object constraints)：基于目标容器的组件方向，返回给定约束位置对应的组件。
- Component getLayoutComponent(Object constraints)：获取使用给定约束添加的组件。
- int getVgap()：返回组件之间的垂直间距。
- void invalidateLayout(Container target)：使布局无效，指示如果布局管理器缓存了信息，则应该将其丢弃。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



- void layoutContainer(Container target)：将边框布局应用到给定的容器。
- Dimension maximumLayoutSize(Container target)：在给出指定目标容器中的组件的前提下，返回此布局的最大尺寸。
- Dimension minimumLayoutSize(Container target)：返回给定容器的最小尺寸。
- Dimension preferredLayoutSize(Container target)：基于容器中的组件，使用此布局管理器确定目标容器的首选大小。
- void removeLayoutComponent(Component comp)：从此边界布局中移除指定组件。
- void setHgap(int hgap)：设置组件之间的水平间距。
- void setVgap(int vgap)：设置组件之间的垂直间距。
- String toString()：返回此边界布局的状态的字符串表示形式。

【范例 10-13】示例代码 10-13 是一个测试边界布局管理器的示例。

示例代码 10-13

```
01  public class NewBorderLayout extends Frame{
02       //创建 5 个按钮，分别代表东、西、南、北、中。
03       Button b1=new Button("东");
04       Button b2=new Button("西");
05       Button b3=new Button("南");
06       Button b4=new Button("北");
07       Button b5=new Button("中");
08
09       NewBorderLayout(){
10            //设置窗口容器的名称
11            this.setTitle("边界布局管理器示例");
12            //设置布局
13            this.setLayout(new BorderLayout());
14            //添加按钮并设置位置
15            this.add(b1,BorderLayout.EAST);
16            this.add(b2,BorderLayout.WEST);
17            this.add(b3,BorderLayout.SOUTH);
18            this.add(b4,BorderLayout.NORTH);
19            this.add(b5,BorderLayout.CENTER);
20            //设置窗口的显示位置以及大小
21            this.setBounds(100, 100, 250, 250);
22            this.setVisible(true);
23       }
24
25  public static void main(String[] args) {
26       new NewBorderLayout();
27  }
28  }
```

【运行结果】程序运行结果如图 10-13 所示。

【代码解析】以上程序中，首先创建类 NewBorderLayout，继承 Frame 类，并创建五个按钮，接下来设置窗口的大小及显示位置。分别将五个按钮添加到窗口中，并设定按钮各自所在的位置。最后在主方法中进行调用显示窗口。



图 10-13　边界布局管理器效果

10.4.4　卡片布局管理器

卡片布局管理器（CardLayout）可以将组件以卡片的形式

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

进行布局。卡片布局管理就像是桌子上面摞了几本书，从上面看的话，只能看到一本。采用卡片布局管理器进行组件的布局，每个组件都属于不同的层次，并且会充满整个容器，也就是组件的大小就是容器的大小，在添加组件的时候，需要指定所在卡片的名称。CardLayout 中提供了一些常用的方法，如下所示。

- void addLayoutComponent(Component comp, Object constraints)：将指定的组件添加到此卡片布局的内部名称表。
- void first(Container parent)：翻转到容器的第一张卡片。
- int getHgap()：获取组件之间的水平间距。
- float getLayoutAlignmentX(Container parent)：返回沿 X 轴的对齐方式。
- float getLayoutAlignmentY(Container parent)：返回沿 Y 轴的对齐方式。
- int getVgap()：获取组件之间的垂直间距。
- void invalidateLayout(Container target)：使布局无效，指示如果布局管理器缓存了信息，则应该将其丢弃。
- void last(Container parent)：翻转到容器的最后一张卡片。
- void layoutContainer(Container parent)：使用此卡片布局布置指定的容器。
- Dimension maximumLayoutSize(Container target)：返回给定容器的最大尺寸。
- Dimension minimumLayoutSize(Container parent)：返回给定容器的最小尺寸。
- void next(Container parent) ：翻转到指定容器的下一张卡片。
- Dimension preferredLayoutSize(Container parent)：使用此卡片布局确定容器参数的首选大小。
- void previous(Container parent)：翻转到指定容器的前一张卡片。
- void removeLayoutComponent(Component comp)：从布局中移除指定的组件。
- void setHgap(int hgap)：设置组件之间的水平间距。
- void setVgap(int vgap)：设置组件之间的垂直间距。
- void show(Container parent, String name)：翻转到使用 addLayoutComponent 添加到此布局的具有指定 name 的组件。
- String toString()：返回此卡片布局状态的字符串表示形式。

【范例 10-14】示例代码 10-14 是一个测试卡片布局管理器的示例。

示例代码 10-14

```
01  public class NewCardLayout {
02      public static void main(String[] args) {
03          //创建 Frame 并设置布局
04          Frame f=new Frame("卡片布局管理器示例");
05          f.setLayout(new CardLayout());
06          //创建两个按钮，并添加到窗口中
07          f.add(new Button("按钮一"),"card1");
08          f.add(new Button("按钮二"),"card2");
09          //设置窗口的显示大小及位置
10          f.setBounds(100, 100, 250, 250);
11          f.setVisible(true);
12      }
13  }
```

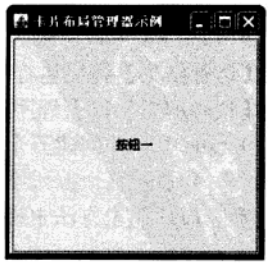


图 10-14　卡片布局管理器效果

【运行结果】程序运行结果如图 10-14 所示。

【代码解析】以上程序中，首先创建一个窗口容器，并设置窗口的布局方式为卡片布局管

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

理器，创建两个按钮分别添加到窗口中，最后设置窗口的显示大小及位置。

10.5　习题

【习题 10-1】编写一个计算器界面，能够进行一些基本的运算，如图 10-15 所示。

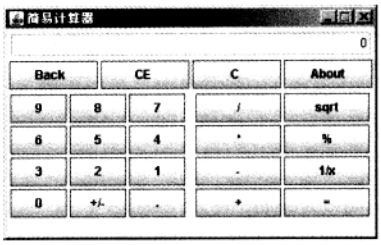


图 10-15　计算器效果

【分析】该题主要考查的是对基本组件和布局知识的掌握。首先要使用面板将窗口分成三个部分，其中一个位于最上方，用于显示输入及输出结果，另两个面板将下面的部分分为两部分，左边部分用于放置显示数字的按钮，右边部分用于放置计算符号。

【核心代码】

```
public class Calculator extends WindowAdapter implements ActionListener {
    JFrame list;
    JTextField show;
    JButton bc, c, ce, ab, jia, jian, cheng, chu, equ, point, sqrt, ds, bfh,
            zf;
    // 按钮 退格，清空，复位，关于，加，减，乘，除，等号，小数点，2 次方根，倒数，百分号，正负号
    JButton b[] = new JButton[10];      // 按钮数组，数字键 0～9
    double sum = 0, getValue;
    int i = 0, j = 0, p = 0, l, action;
    JDialog about;
    final int slength = 30;             // 设置结果显示有效长度
    public void disp() {
        list = new JFrame("简易计算器");
        ……                              //此处省略的是对 JFrame 属性进行配置的代码
        list.setResizable(false);
        show = new JTextField(31);
        ……                              //此处省略的是对 JTextField 组件进行设置的代码
        show.setEditable(false);
        list.add(show);
        Panel dispTop = new Panel();
        list.add(dispTop);
        dispTop.setLayout(new GridLayout(1, 4, 3, 3));
        bc = new JButton("  Back    ");
        bc.setForeground(Color.BLUE);
        dispTop.add(bc);
        ……          //此处省略添加计算器所有的数字键按扭以及为各按扭添加监听事件的代码
    }
    public void actionPerformed(ActionEvent e) {
        getValue = Double.valueOf(show.getText()).doubleValue();
        if (e.getSource() == jia) {     // 加运算，可连加
            if (j == 0) {
                sum = getValue;
            } else if (action == 1) {
                sum += getValue;
            }
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
            setSum();
            j++;
            p = 0;
            i = 0;
            action = 1;
        } else if (e.getSource() == jian) {          // 减运算，可连减
            if (j == 0) {
                sum = getValue;
            } else if (action == 2) {
                sum -= getValue;
            }
            setSum();
            j++;
            p = 0;
            i = 0;
            action = 2;
        } else if (e.getSource() == cheng) {         // 乘运算，可连乘
            if (j == 0) {
                sum = getValue;
            } else if (action == 3) {
                sum *= getValue;
            }
            setSum();
            j++;
            p = 0;
            i = 0;
            action = 3;
        } else if (e.getSource() == chu) {           // 除运算，可连除
            if (j == 0)
                sum = getValue;
            else if (action == 4) {
                sum /= getValue;
            }
            setSum();
            j++;
            p = 0;
            i = 0;
            action = 4;
        } else if (e.getSource() == equ) {           // 等号,运算最后一个操作数
            switch (action) {
            case 1:
                show.setText(String.valueOf(sum += getValue));
                break;
            case 2:
                show.setText(String.valueOf(sum -= getValue));
                break;
            case 3:
                show.setText(String.valueOf(sum *= getValue));
                break;
            case 4:
                show.setText(String.valueOf(sum /= getValue));
                break;
            }
            setSum();
            i = 0;
            j = 0;
            action = 0;
        } else if (e.getSource() == point) {         // 小数点,只能按一个小数点
            if (p == 0)
                show.setText(show.getText() + e.getActionCommand());
            p = 1;
        } else if (e.getSource() == c || e.getSource() == ce) {  // 清空与复位
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
            i = 0;
            j = 0;
            p = 0;
            sum = 0;
            action = 0;
            show.setText("0");
        ... ...                                          //此处省略一部分计算代码
        for (l = 0; l < 10; l++) {                       // 0~9 数字键触发
            if (e.getSource() == b[l]) {
                if (i == 0)
                    show.setText("");
                String s = show.getText();
                if (s.length() < slength)
                    show.setText(show.getText() + e.getActionCommand());
                if (e.getSource() == b[0] && getValue == 0 && p == 0)
                    show.setText("0");
                if (e.getSource() != b[0] && getValue == 0 && p == 0)
                    show.setText(e.getActionCommand());
                i++;                                     // i 用来标记数字键触发的状态
            }
        }
    }
    public void setSum() {                               // 把计算结果显示出来
        show.setText(String.valueOf(sum));
        String s = show.getText();
        char a = s.charAt((s.length() - 1));
        char b = s.charAt((s.length() - 2));
        if (a == '0' && b == '.') {                  // 如果是整数,则去掉后面的小数点和 0
            show.setText(String.valueOf(Math.round(sum)));
        }
    }
    public void windowClosing(WindowEvent e) {  //为窗口上的按扭添加事件
        if (e.getSource() == about)
            about.setVisible(false);
        else if (e.getSource() == list)
            System.exit(0);
    }
```

【习题 10-2】编写一个电话本程序界面，要求完成主要的界面布局，显示联系人姓名、电话、邮箱信息，并在最后设置一个按钮，用于信息录入，如图 10-16 所示。

【分析】本题主要考查的是对基本控件的使用，其中主要使用到了按钮、文本框和标签。文本框用于获取用户输入的信息，标签用于提示用户需要输入的内容，单击按钮则提交。

【核心代码】

```
class InputArea extends Panel
implements ActionListener
{ File f=null;
  RandomAccessFile out;
  Box baseBox ,boxV1,boxV2;
  TextField name,email,phone;
  Button button;
  InputArea(File f)
  {   setBackground(Color.cyan);
      this.f=f;
      name=new TextField(12);
      email=new TextField(12);
      phone=new TextField(12);
      button=new Button("录入");
      button.addActionListener(this);
      boxV1=Box.createVerticalBox();
      boxV1.add(new Label("输入姓名"));
```

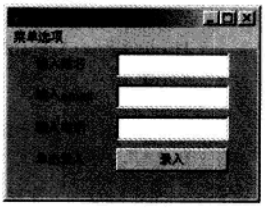


图 10-16　电话本效果

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
        boxV1.add(Box.createVerticalStrut(8));
        boxV1.add(new Label("输入 email"));
        boxV1.add(Box.createVerticalStrut(8));
        boxV1.add(new Label("输入电话"));
        boxV1.add(Box.createVerticalStrut(8));
        boxV1.add(new Label("单击录入"));
        boxV2=Box.createVerticalBox();
        boxV2.add(name);
        boxV2.add(Box.createVerticalStrut(8));
        boxV2.add(email);
        boxV2.add(Box.createVerticalStrut(8));
        boxV2.add(phone);
        boxV2.add(Box.createVerticalStrut(8));
        boxV2.add(button);
        baseBox=Box.createHorizontalBox();
        baseBox.add(boxV1);
        baseBox.add(Box.createHorizontalStrut(10));
        baseBox.add(boxV2);
        add(baseBox);
    }
    public void actionPerformed(ActionEvent e)
    { try{
          RandomAccessFile out=new RandomAccessFile(f,"rw");
          if(f.exists())
            { long length=f.length();
              out.seek(length);
            }
          out.writeUTF("姓名:"+name.getText());
          out.writeUTF("eamil:"+email.getText());
          out.writeUTF("电话:"+phone.getText());
          out.close();
        }
       catch(IOException ee){}
    }
}

class CommFrame extends Frame implements ActionListener
{
    String fileName="通讯录.txt";//文件名
    File file=null;              //文件对象
    MenuBar bar;                 //菜单吧
    Menu fileMenu;
    MenuItem 录入,显示,查找,删除;
    TextArea show;               //显示面板
    InputArea inputMessage;      //录入面板
    Find t;                      //查找面板
    Delete d;                    //删除面板
    CardLayout card=null;        //卡片式布局.
    Panel pCenter;
    CommFrame()
    {   //初始化各个面板
     ... ...                                    //此处省略一部分配置代码
        //添加布局管理器
        card=new CardLayout();
        pCenter=new Panel();
        pCenter.setLayout(card);
        //给各个面板注册响应对象
        pCenter.add("录入",inputMessage);
        pCenter.add("显示",show);
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



```
        pCenter.add("查找",t);
        pCenter.add("删除",d);
        add(pCenter,BorderLayout.CENTER);
        //关闭程序
        addWindowListener(new WindowAdapter()
                  { public void windowClosing(WindowEvent e)
                      { System.exit(0);
                      }
                  });
       setVisible(true);                  //设窗口为可见
       setBounds(100,50,420,380);         //设置窗口大小
       validate();                        //更新窗口
    }
    public void actionPerformed(ActionEvent e)          //监听器类 监听菜单栏
    {   if(e.getSource()==录入)            //录入操作
        { card.show(pCenter,"录入");
        }
      else if(e.getSource()==显示)         //显示记录操作
        { int number=1;
          show.setText(null);
          card.show(pCenter,"显示");
         try{ RandomAccessFile in=new RandomAccessFile(file,"r");
              String 姓名=null;
              while((姓名=in.readUTF())!=null)          //循环读取记录
                 { show.append("\n"+number+" "+姓名);
                   show.append("\n"+in.readUTF());      //读取 email.
                   show.append("\n"+in.readUTF());      //读取 phone
                   show.append("\n-------------------- ");
                   number++;
                 }
              in.close();
            }
          catch(Exception ee){}
        }
       else if((e.getSource()==查找)){//查找操作
      card.show(pCenter,"查找");
       }
       else if((e.getSource()==删除)){
      card.show(pCenter,"删除");
       }
    }
```

【习题 10-3】编写一个程序，创建一个窗口，并在窗口中设置标签等控件，并将这些控件按照顺序布局管理器的格式进行布局。

【分析】该题主要考查的是对顺序布局管理器的使用。

【核心代码】

```
NewFlowLayout()
     {
     //设置窗口名称
     this.setTitle("FlowLayout 布局管理器");
     //设置布局管理器为 FlowLayout
     this.setLayout(new FlowLayout());
     //添加按钮
     this.add(b1);
     this.add(b2);
this.add(b3);
     this.add(b4);
     //设置窗口大小及显示位置
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
        this.setBounds(100,100,250,250);
        //设置窗口可见
        this.setVisible(true);
    }
```

【习题 10-4】编写一个选课程序，要求完成主界面，并显示几门课程的复选框，最后设置一个按钮。

【分析】本题主要考查的是对基本控件的使用。

【核心代码】

```
public class SelectCourse {
    public static void main(String[] args) {
        Frame f = new Frame("选课");
        f.setLayout(new GridLayout(2,1));
        Checkbox c1 = new Checkbox("组织行为学");
        Checkbox c2 = new Checkbox("跨国公司管理");
        Checkbox c3 = new Checkbox("营销管理学");
        Checkbox c4 = new Checkbox("管理经济学");
        f.add(c1);
        f.add(c2);
        f.add(c3);
        f.add(c4);
        f.add(new Button("提交"));
        f.setSize(300, 200);
        f.setVisible(true);
    }
}
```

【习题 10-5】编写一个文本编辑器程序，要求在窗口的上部显示几个按钮，下面是文本编辑区。

【分析】本题主要考查的是对基本控件的使用。

【核心代码】

```
public class EditText {
    public static void main(String[] args) {
        Frame f = new Frame("文本编辑器");
        f.setLayout(new BorderLayout());
        Panel t = new Panel();
        t.setLayout(new FlowLayout());
        t.add(new Button("File"));
        t.add(new Button("Edit"));
        t.add(new Button("Format"));
        t.add(new Button("view"));
        t.add(new Button("Help"));
        f.add(t, BorderLayout.NORTH);
        Panel a = new Panel();
        a.add(new TextArea());
        f.add(a, BorderLayout.CENTER);
        f.setSize(600, 400);
        f.setVisible(true);
    }
}
```

【习题 10-6】编写一个登记程序，首先输入姓名，然后在一个下拉列表中选择国籍，最后有个提交按钮。

【分析】本题主要考查的是对基本控件的使用。

【核心代码】

```
public class CheckIn {
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



```
    public static void main(String[] args) {
        Frame f = new Frame("入住登记");
        f.setLayout(new GridLayout(3, 1));
        Panel p1 = new Panel();
        p1.add(new Label("姓名："));
        p1.add(new TextField(20));
        Panel p2 = new Panel();
        p2.add(new Label("国籍："));
        List l = new List(4);
        l.add("中国");
        l.add("美国");
        l.add("英国");
        l.add("法国");
        p2.add(l);

        Panel p3 = new Panel();
        p3.add(new Button("提交"));

        f.add(p1);
        f.add(p2);
        f.add(p3);

        f.setSize(400, 300);
        f.setVisible(true);
    }
}
```

【习题 10-7】实现一个主界面，使用网格布局管理器对九个按钮进行布局。

【分析】该题主要考查的是对网格布局管理器的使用。

【核心代码】

```
public class GridLayoutDemo {
    public static void main(String[] args) {
        Frame f = new Frame("网格布局器");
        f.setLayout(new GridLayout(3, 3));
        for (int i = 1; i < 10; i++) {
            f.add(new Button(""+i));
        }
        f.setSize(400, 300);
        f.setVisible(true);
    }
}
```

【习题 10-8】实现一个主界面，使用顺序布局管理器，对四个 TextArea 进行布局。

【分析】该题主要考查的是对顺序布局管理器的使用。

【核心代码】

```
public class FlowLayoutDemo {
    public static void main(String[] args) {
        Frame f = new Frame("顺序布局器");
        f.setLayout(new FlowLayout());
        for (int i = 1; i < 5; i++) {
            f.add(new TextArea(10,10));
        }
        f.setSize(400, 300);
        f.setVisible(true);
    }
}
```

【习题 10-9】实现一个主界面，使用边界布局器，对五个按钮进行布局。

【分析】该题主要考查的是对边界布局管理器的使用。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

【核心代码】

```
public class BorderLayoutDemo {
    public static void main(String[] args) {
        Frame f = new Frame("边界布局器");
        f.setLayout(new BorderLayout());

        f.add(new Button("North"), BorderLayout.NORTH);
        f.add(new Button("South"), BorderLayout.SOUTH);
        f.add(new Button("East"), BorderLayout.EAST);
        f.add(new Button("West"), BorderLayout.WEST);
        f.add(new Button("Center"), BorderLayout.CENTER);


        f.setSize(400, 300);
        f.setVisible(true);
    }
}
```

【习题 10-10】实现一个主界面，使用卡片布局管理器对四个 Button 进行布局。

【分析】该题主要考查的是对卡片布局管理器的使用。

【核心代码】

```
public class CardLayoutDemo {
    public static void main(String[] args) {
        final Frame f = new Frame("卡片布局器");
        final CardLayout c = new CardLayout(5, 10);
        f.setLayout(c);
        ActionListener listener = new ActionListener(){
            public void actionPerformed(ActionEvent e) {
                c.next(f);
            }
        };
        for (int i = 1; i < 5; i++) {
            Button b = new Button("单击切换"+i);
            b.addActionListener(listener);
            f.add(b, "" +i);
        }
        f.setSize(400, 300);
        f.setVisible(true);
    }
}
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第 11 章　Swing

Swing 元素的屏幕显示性能要比 AWT 要好，而且 Swing 是使用纯 Java 来实现的，所以 Swing 也理所当然地具有 Java 的跨平台性。但 Swing 并不是真正使用原生平台提供设备，而是仅仅在模仿，因此，可以在任何平台上使用 Swing 图形用户界面组件。它不必在自己本地窗口中绘制组件，而是在所在的重量级窗口中绘制，因为 Swing 绝大部分是轻量级的组件。



11.1　Swing 基础

Swing 解决了 AWT 中出现的一些问题，可以说是 AWT 的扩展，提供了支持跨平台功能更加丰富的组件库，这好比是一个系列的汽车，每年的款式都会比前一年的款式增加一些功能，丰富一些内饰。Swing 中与 AWT 相对应的组件使用方法基本相同，只是 Swing 采用了轻量级的方法在 AWT 的基础上重新编写了该组件，而且也增加了一些 AWT 中所没有的组件。

Javax.swing 包中有顶层容器和轻量级两种类型的组件，Swing 轻量级的组件都是由 AWT 的 Container 类直接或者间接派生而来的。

- Com.sum.swing.plaf.motif：实现 Motif 界面样式代表类。
- Com.sum.java.swing.plaf.windows：实现 Windows 界面样式的代表类。
- javax.swing：Swing 组件和使用工具。
- javax.swing.border：Swing 轻量组件的边框。
- javax.swing.colorchooser：JcolorChooser 的支持类/接口。
- javax.swing.event：事件和侦听器类。
- javax.swing.filechooser：JFileChooser 的支持类/接口。
- javax.swing.pending：未完全实现的 Swing 组件。
- javax.swing.plaf：抽象类，定义 UI 代表的行为。
- javax.swing.plaf.basic：实现所有标准界面样式公共基类。
- javax.swing.plaf.metal：实现 Metal 界面样式代表类。
- javax.swing.table：Jtable 组件。
- javax.swing.text：支持文档的显示和编辑。
- javax.swing.text.html：支持显示和编辑 HTML 文档。
- javax.swing.text.html.parser：Html 文档的分析器。
- javax.swing.text.rtf：支持显示和编辑 RTF 文件。
- javax.swing.tree：Jtree 组件的支持类。
- javax.swing.undo：支持取消操作。



11.2　Swing 组件

Swing 中的组件和 AWT 中的有些类似，使用方法基本相同。但 Swing 又为每一个组件增

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

添了新的方法，并提供了更多的高级组件。Swing 中的组件通常是以“J”开头的，如 JButton、JLabel、JFrame、JPanel 等。本节将从 Swing 的基本组件中选取几个比较典型的组件进行详细讲解。

11.2.1 按钮

Swing 中的按钮是 Jbutton，它是 javax.swing.AbstractButton 类的子类，Swing 中的按钮可以显示图像，并且可以将按钮设置为窗口的默认图标，而且还可以将多个图像指定给一个按钮。在前面的 HelloWorld 例子中就用到了一个按钮。在 Jbutton 中有如下几个比较常用的构造方法。

- JButton(Icon icon) ：按钮上显示图标。
- JButton(String text)：按钮上显示字符。
- JButton(String text, Icon icon)：按钮上既显示图标又显示字符。

【范例 11-1】示例代码 11-1 是一个测试 Jbutton 的程序。

示例代码 11-1

```
01      public class JButtontest {
02          public static void main(String[] args) {
03          JFrame f = new JFrame("这是一个按钮");
04          Container contentPane = f.getContentPane();
05          JButton b = new JButton("按钮");//创建一个带初始文本的按钮。
06          //如果没有设置文字的位置，系统默认值会将文字置于图形的右边中间位置。
07      //设置文本相对于图标的水平方向的位置。
08      b.setHorizontalTextPosition(JButton.CENTER);
09      //设置文本相对于图标的垂直方向的位置。
10      b.setVerticalTextPosition(JButton.BOTTOM);
11          contentPane.add(b);
12          f.pack();
13          f.show();
14          f.addWindowListener(new WindowAdapter() {
15              public void windowClosing(WindowEvent e) {
16              System.exit(0);
17              }
18          });
19      }
20      }
```

【运行结果】程序运行结果如图 11-1 所示。

【代码解析】首先创建一个窗口，并设置窗口的显示文本。然后在窗口中设置一个按钮并设置按钮的显示文字。接下来对文本相对于图标的方向位置进行设置。最后为按钮添加事件监听。



图 11-1 按钮示例效果

11.2.2 复选框

复选框组件是可以完成多项选择的组件，Swing 中的复选框和 AWT 中的复选框相比，优点是在 Swing 复选框中可以添加图片。该类是 javax.swing.JToggleButton 的子类，复选框的构造方法如下。

- JCheckBox()：创建一个无文本、无图标、未被选定的复选框。
- JCheckBox(Action a)：创建一个复选框，属性由参数 Action 提供。
- JCheckBox(Icon icon)：创建一个有图标，但未被选定的复选框。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

- JCheckBox(Icon icon, boolean selected)：创建一个有图标，并且指定是否被选定的复选框。
- JCheckBox(String text)：创建一个有文本，但未被选定的复选框。
- JCheckBox(String text, boolean selected)：创建一个有文本，并且指定是否被选定的复选框。
- JCheckBox(String text, Icon icon)：创建一个指定文本和图标，但未被选定的复选框。
- JCheckBox(String text, Icon icon, boolean selected)：创建一个指定文本和图标，并指定是否被选定的复选框。

【范例 11-2】示例代码 11-2 是一个测试复选框的程序。

示例代码 11-2

```
01  public class JCheckBoxTest  {
02      JFrame f = null;
03      JCheckBoxTest() {
04          f = new JFrame("复选框示例");                    // 创建一个 JFrame 实例对象
05          Container contentPane = f.getContentPane(); // 定义一个容器
06          contentPane.setLayout(new GridLayout(2, 1));     // 设置窗口的布局
07          JPanel p1 = new JPanel();                        // 创建一个面板对象 p1
08          p1.setLayout(new GridLayout(1, 3));
09          p1.setBorder(BorderFactory.createTitledBorder("选择你想要去的城市?
10  "));
11          JCheckBox c1 = new JCheckBox("北京");
12          JCheckBox c2 = new JCheckBox("深圳");
13          JCheckBox c3 = new JCheckBox("青岛");
14          p1.add(c1);
15          p1.add(c2);
16          p1.add(c3);
17
18          contentPane.add(p1);
19
20          f.pack();
21          f.show();
22          f.addWindowListener(new WindowAdapter() {        // 创建一个窗口监听器
23                      public void windowClosing(WindowEvent e) {
24                          System.exit(0);
25                      }
26                  });
27      }
28      public static void main(String args[]) {
29          new JCheckBoxTest();
30      }
31
32  }
```

【运行结果】程序运行结果如图 11-2 所示。

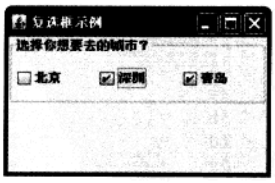


图 11-2　复选框示例效果

【代码解析】首先要创建 JFrame 对象、容器，并设置窗口的布局格式，定义三个 JCheckBox 对象，并设置各自的显示文本。然后为窗口设置事件监听。最后创建主方法，并在主方法中调用构造方法。

11.2.3　单选按钮

单选按钮（JradioButton）与 AWT 中的单选按钮组件功能类似，通常 JRadioButton 和

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

ButtonGroup 配合一起使用，作用就是一次创建一组按钮，并且在这一组按钮中，每一次只能够选中一个按钮。其中，需要使用 add()方法将 JRadioButton 添加到 ButtonGroup 中。这一点和 AWT 中的单选按钮相类似，就像是购物后选择赠品一样，赠品摆放了一堆，但是只能够选择一个。JRadioButton 的常用构造方法如下。

- JRadioButton()：用于创建一个未指定图标和文本，并且未被选定的单选按钮。
- JRadioButton(Action a)：用于创建一个属性来自 Action 的单选按钮。
- JRadioButton(Icon icon)：用于创建一个指定图标，未被选定的单选按钮。
- JRadioButton(Icon icon, boolean selected)：用于创建一个指定图像和状态的单选按钮。
- JRadioButton(String text)：用于创建一个指定文本，未被选定的单选按钮。
- JRadioButton(String text, boolean selected)：用于创建一个指定文本和选择状态的单选按钮。
- JRadioButton(String text, Icon icon)：用于创建一个指定文本和图标，未被选定的单选按钮。
- JRadioButton(String text, Icon icon, boolean selected)：用于创建一个具有指定的文本、图像和选择状态的单选按钮。

【范例 11-3】示例代码 11-3 是一个测试单选按钮的程序。

示例代码 11-3

```
01      public class JRadioButtonTest
02      {
03          JFrame f = null;
04          JRadioButton r4 = null;
05          JRadioButton r5 = null;
06          JRadioButtonTest() {
07              f = new JFrame("单选框示例");              // 创建一个 JFrame 的对象
08              Container contentPane = f.getContentPane(); // 创建一个容器
09              contentPane.setLayout(new GridLayout(2, 1));// 设置这个窗口的布局
10              JPanel p1 = new JPanel();                  // 创建一个面板对象 p1
11              p1.setLayout(new GridLayout(1, 3));        // 设置布局管理器的格式
12              p1.setBorder(BorderFactory.createTitledBorder("选择你喜欢的城市
13  "));
14              // 定义 3 个 JRadioButton 单选按钮
15              JRadioButton r1 = new JRadioButton("北京");
16              JRadioButton r2 = new JRadioButton("上海");
17              JRadioButton r3 = new JRadioButton("青岛");
18              p1.add(r1);
19              p1.add(r2);
20              p1.add(r3);
21              JPanel p2 = new JPanel();
22              p2.setLayout(new GridLayout(2, 1));
23              contentPane.add(p1);
24              contentPane.add(p2);
25              f.pack();
26              f.show();
27              f.addWindowListener(new WindowAdapter() {// 添加一个窗口监听器
28                          public void windowClosing(WindowEvent e) {
29                              System.exit(0);
30                          }
31                      });
32          }
33          public static void main(String args[]) {
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
34                  new JRadioButtonTest();
35              }
    }
```

【运行结果】程序运行结果如图 11-3 所示。

【代码解析】以上程序中，首先是创建容器，设置容器布局格式，然后再创建三个单选按钮，将三个单选按钮分别加入到面板中。

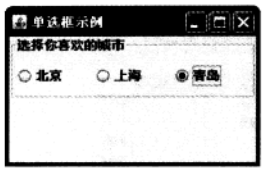


图 11-3　单选按钮示例效果

11.2.4　组合框

组合框，顾名思义，就是将一些组件（如按钮或下拉菜单等）进行组合的组件。用户可以使用下拉菜单选择不同的选项，如果此时的组合框处于编辑状态，还可以在组合框中输入值。

组合框常用的构造方法如下。

- JComboBox()：创建一个没有数据选项的组合框。
- JComboBox(ComboBoxModel aModel)：创建一个数据来源于 ComboBoxModel 的组合框。
- JComboBox(Object[] items)：创建一个以指定数组元素作为选项的组合框。
- JComboBox(Vector<?> items)：创建一个指定 Vector 中元素的组合框。

【范例 11-4】示例代码 11-4 是一个测试组合框的程序。

示例代码 11-4

```
01  public class JComboBoxTest {
02  public static void main(String[] args) {
03       JFrame jf = new JFrame("JComboBoxDemo1");          // 创建一个 JFrame 对象
04       Container contentPane = jf.getContentPane();       // 定义一个容器
05       contentPane.setLayout(new GridLayout(1, 2));
06       // 定义一个字符串数组，并将其初始化
07       String[] str = { "故宫", "泰山", "张家界", "颐和园", "孔府"};
08       Vector vector = new Vector();                      //创建一个 Vector 对象
09       //向 Vector 对象中添加数据
10       vector.addElement("北京");
11       vector.addElement("上海");
12       vector.addElement("青岛");
13       vector.addElement("广州");
14       vector.addElement("成都");
15       vector.addElement("西安");
16       JComboBox combo1 = new JComboBox(str);        // 定义一个 JComboBox17 对象
17       // 利用 JComboBox 类所提供的 addItem()方法，加入一个项目到此 JComboBox19 中。
18       combo1.addItem("泰山");
19       //创建一个带有指定标题的标题框
20  combo1.setBorder(BorderFactory.createTitledBorder("你想去哪个景点游玩?
21  "));
22            JComboBox combo2 = new JComboBox(vector);
23            combo2.setBorder(BorderFactory.createTitledBorder("你喜欢的城市"));
24            contentPane.add(combo1);
25            contentPane.add(combo2);
26            jf.pack();
27            jf.show();
28  jf.addWindowListener(new WindowAdapter() {        // 添加窗口监听器
29                       public void windowClosing(WindowEvent e) {
30                            System.exit(0);
31                       }
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
32                      });
33          }
}
```

【运行结果】程序运行结果如图 11-4 所示。

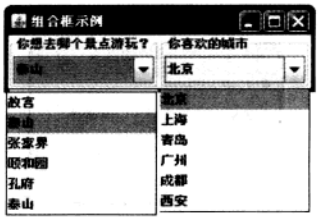


图 11-4　组合框效果

11.2.5　进度条

进度条就是用图形的方式来模拟描述任务进度的组件。任务进行的过程中，进度条会以百分比的形式来显示。在程序结束后，进度条会显示百分之百。进度条的常用构造方法如下。

- JProgressBar()：创建一个显示矩形的水平进度条。
- JProgressBar(BoundedRangeModel newModel)：创建一个指定进度条数据模型的水平进度条。
- JProgressBar(int orient)：创建一个指定方向（VERTICAL 或 HORIZONTAL）的进度条。
- JProgressBar(int min, int max)：创建一个指定最小值和最大值的水平进度条。
- JProgressBar(int orient, int min, int max)：创建一个指定方向、最小值和最大值的进度条。

11.2.6　表格

表格（Jtable）是 Swing 新增加的组件，主要是为了将数据以表格的形式显示。通常用数据模型类的对象来保存数据，数据模型类派生于 AbstractTableModel 类，并且必须重写抽象模型类的几个方法，如 getColumnCount、getRowCount、getColumnName、和 getValueAt。因为表格会从这个数据模型类的对象中自动获取数据，数据模型类的对象负责表格大小（行/列）、数据填写、表格单元格更新等与表格有关的属性和操作。表格的构造方法如下。

- JTable()：使用系统默认的模型创建一个 JTable 实例。
- JTable(int numRows,int numColumns)：使用 DefaultTableModel 构造具有 numRows 行和 numColumns 列个空单元格的 JTable。
- JTable(Object[][] rowData,Object[][] columnNames)：创建一个显示二维数据的表格。
- JTable(TableModel dm)：创建一个指定数据模式和默认字段模式的 JTable 实例。
- JTable(TableModel dm,TableColumnModel cm)：创建一个指定数据模式和字段模式的 JTable 实例。
- JTable(TableModel dm,TableColumnModel cm,ListSelectionModel sm)：创建一个指定数据模式、字段模式与选择模式的 JTable 实例。
- JTable(Vector rowData,Vector columnNames)：创建一个以 Vector 为数据源，并显示列名称的 JTable 实例。

11.2.7　树

树（Jtree）中特定的节点可以由 TreePath 标识或由其显示行标识。当展开某一个节点的所有祖先时，将显示该节点，折叠节点是位于折叠祖先下面的节点。可查看节点的父类节点都是可以展开的，但是它们可以显示也可以不显示。显示节点必须是可查看的并且位于显示区域。树相关的方法如下。

- isRootVisible()：返回树的根节点是否显示。
- isVisible()：返回当前路径查看标识。
- makeVisible()：设置当前路径的查看标识。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

- setRootVisible()：设置是否显示树的根节点。
- scrollPathToVisible()：确保展开所有的路径。
- scrollRowToVisible()：按行滚动标识的条目，直到显示出来。
- getVisibleRowCount()：返回显示区域的显示行的数目。
- setVisibleRowCount()：设置显示区域中显示行的数目。

树的构造方法如下。

- JTree()：创建一个空节点的树。
- JTree(Hashtable<?,?> value)：创建一个由 Hashtable 元素构成节点的 JTree，它不显示根。
- JTree(Object[] value)：创建一个指定数组的元素作为不给显示的新的根节点的子节点的树实例。
- JTree(TreeModel newModel)：创建一个使用指定数据模型，并显示根节点的树实例。
- JTree(TreeNode root)：创建一个指定 TreeNode 作为其根，并显示根节点的树实例。
- JTree(TreeNode root, boolean asksAllowsChildren)：创建一个指定 TreeNode 作为其根，并指定根节点的显示方式的树实例。

【范例 11-5】示例代码 11-5 是一个演示树的程序。

示例代码 11-5

```
01  public class JTreeTest {
02      public static void main(String args[]) {
03          JFrame f = new JFrame("树示例");                    //创建窗体。
04          //设置用户在此窗体上发起 "close" 时默认执行的操作
05          f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
06          f.getContentPane();
07          Box box = Box.createHorizontalBox();               //创建 Box 类对象
08          JTree tree1 = new JTree();                         //创建树。
09  tree1.putClientProperty("JTree.lineStyle", "Angled");//向此组件添加任意的键/值
10          JScrollPane scrollPane1 = new JScrollPane(tree1);//创建一个滚动面板
11          tree1.setAutoscrolls(true);
12      JTree tree2 = new JTree();
13      JScrollPane scrollPane2 = new JScrollPane(tree2);
14      //向 Box 容器添加滚动面板
15      box.add(scrollPane1, BorderLayout.WEST);
16      box.add(scrollPane2, BorderLayout.EAST);
17      f.getContentPane().add(box, BorderLayout.CENTER);
18      f.setSize(300, 240);
19      f.setVisible(true);
20
21  }
```

【运行结果】程序运行结果如图 11-5 所示。



图 11-5　树示例效果图

11.2.8　文本框

文本框具有文本输入和编辑的功能，文本框组件用于获取用户输入的文本。除此之外，还有文本区，文本区和文本框的区别是，前者可以输入多行文本，而文本框只接受单行文本的输入。

实现文本框功能的类是 JTextField，其中提供了多个方法，例如，可以设置输入的文本字符的长度限制。密码框和文本框的外观一样，并且也继承自 JTextField 类，密码框只提供专门的密码输入，输入内容不能直接

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

显示，在密码框中以星号或其他形式的符号显示。

【范例 11-6】示例代码 11-6 是一个测试文本框的程序。

示例代码 11-6

```
01  public class JTextFieldDemo {
02  public static void main(String args[]) {
03  JFrame jframe = new JFrame("文本框和密码框示例");
04      Container contentPane = jframe.getContentPane();      // 返回窗体容器。
05      contentPane.setLayout(new BorderLayout());            // 设置窗体容器布局。
06      JPanel jpanel = new JPanel();                         // 声明面板容器。
07      jpanel.setLayout(new GridLayout(3, 2));               // 设置面板布局。
08      jpanel.setBorder(BorderFactory.createTitledBorder("请输入你的登陆信息"));
09      // 设置面板边界。
10      JLabel lable1 = new JLabel("用户名: ", JLabel.CENTER); // 创建标签。
11      JLabel lable2 = new JLabel("密码: ", JLabel.CENTER);
12      JTextField t1 = new JTextField(10);          // 创建单行文本框，其长度为 10。
13      JPasswordField t2 = new JPasswordField(10);
14      jpanel.add(lable1);        // 将标签及单行文本框依次添加到面板中。
15      jpanel.add(t1);
16      jpanel.add(lable2);
17      jpanel.add(t2);
18      contentPane.add(jpanel);
19      jframe.setSize(500, 200);
20      jframe.setVisible(true);
21      jframe.addWindowListener(new WindowAdapter()     // 关闭事件。
22              {
23                  public void windowClosing(WindowEvent e) {
24                      System.exit(0);
25                  }
26              });
27  }
28  }
```

【运行结果】程序运行结果如图 11-6 所示。

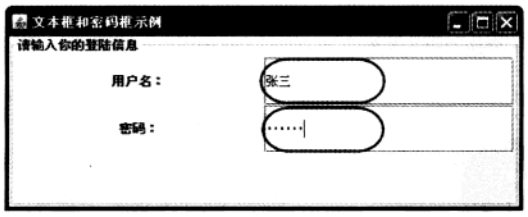


图 11-6　文本框示例效果

在文本框中只能够输入单行的文字，而在文本区中可以输入多行文字，该组件是由类 JTextArea 负责的。

【范例 11-7】示例代码 11-7 是一个测试文本区的程序。

示例代码 11-7

```
01  public class TextAreaTest {
02  public static void main(String args[]) {
03      JFrame jframe = new JFrame("文本区示例");
04      Container contentPane = jframe.getContentPane();      // 返回窗体容器。
05      contentPane.setLayout(new BorderLayout());            // 设置窗体容器布局。
06      JPanel jpanel = new JPanel();                         // 声明面板容器。
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
07        JTextArea jta = new JTextArea(10,20);        // 创建单行文本框，其长度为 10。
08        jpanel.add(jta);
09        contentPane.add(jpanel);
10        jframe.setSize(500, 200);
11        jframe.setVisible(true);
12        jframe.addWindowListener(new WindowAdapter()     // 关闭事件。
13                {
14                    public void windowClosing(WindowEvent e) {
15                    System.exit(0);
16                    }
17                });
18        }
19    }
```

【运行结果】程序运行效果如图 11-7 所示。

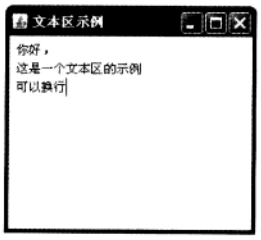


图 11-7　文本区示例效果

11.3　Swing 常用窗体

Swing 常用窗体有 JFrame 和 JDialog 两种。Swing 中的窗体类似于 AWT 中的窗体。

11.3.1　JFrame 窗体

Java 中的顶层窗体被称做是框架，这个框架对应着 AWT 中的 Frame，在 Swing 中对应的则是 JFrame。大多数的标准图形应用程序都会将组件放置到框架中。JFrame 是不通过绘制的方式显示的 Swing 组件。

如果要创建一个 JFrame 窗体有两种方法，一种是创建一个 JFrame 的实例，另一种就是继承 JFrame。JFrame 和 Frame 类似，新创建的窗体也是不可见的，所以还要将 JFrame 的可见属性设置为 true。JFrame 类提供了多个构造方法，可以根据实际需要调用不同的构造方法。

- public JFrame()throws HeadlessException：该方法创建一个初始时候不可见的窗体。
- public JFrame(String title)throws HeadlessException：创建一个标题为 title 的窗体，该窗体在初始的时候也不可见。
- public JFrame(GraphicsConfiguration gc)：以指定的屏幕和空白的标题创建一个窗体，该窗体在初始的时候不可见。
- public JFrame(String title, GraphicsConfiguration gc)：以指定的标题和指定的屏幕创建一个窗体，窗体初始的时候不可见。

JFrame 通过继承父类，也提供了一些比较常用的方法，如下所示。

- boolean isVisible()：该方法的作用是测试窗口是否可见。
- void setVisible(boolean b)：该方法的作用是根据 b 的取值来显示或隐藏窗体。
- boolean isShowing()：该方法的作用是检查组件是否显示在屏幕上，当正在显示并显示在一个显示的容器的时候，返回 true。
- boolean isEnable()：该方法的作用是检查组件是否被激活，当被激活的时候，才能处理各种事件。
- void setEnable(boolean b)：该方法的作用是根据 b 的值来激活或禁用一个组件。
- Point getLocation()：该方法的作用是返回该组件左上角的坐标值，将该坐标值封装在一个 Point 对象中。
- Point getLocationOnScreen()：该方法的作用是返回该组件左上角的坐标值，使用屏幕

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

坐标。

- void setBounds(int x,int y,int width,int height)：该方法作用是设置组件的大小，左上角由 x 和 y 指定坐标而 width 和 height 指定组件的宽度和高度。
- void setLocation(int x,int y)：该方法的作用是重定位组件的坐标。
- void setLocation(Point p)：该方法的作用是把组件重定位，组件的位置由 Point 指定。
- Dimension getSize()：该方法的作用是获得组件的大小。
- void setSize(int width,int height)：该方法的作用是重新设置组件的大小。
- void setSize(Dimension d)：该方法的作用是重设组件大小。
- void toFront()：该方法的作用是该窗体显示到其他任何窗体的前面。
- void toBack()：如果此窗口是可视的，则将此窗口置于后方，如果它是焦点窗口或活动窗口，则会导致丢失焦点或活动状态。
- void setResizable(boolean b)：该方法的作用是决定用户是否可以缩放窗体。
- void setTitle(String s)：该方法的作用是将窗体的标题设为 s。
- void setIconImage(Image image)：该方法的作用是把图像对象用做窗体图标。
- void setUndecorated(boolean b)：该方法的作用是清除对窗体的修饰。
- boolean isUndecorated()：该方法的作用是如果窗体没有修饰的话，返回 true。
- int getExtendedState()：该方法的作用是获得窗体状态。
- void setExtendedState(int state)：该方法的作用是设置窗体状态。

【范例 11-8】示例代码 11-8 是一个测试 Jframe 窗体的程序。

示例代码 11-8

```
01  public class JFrameTest extends JFrame{
02      public JFrameTest(){
03          setSize(new Dimension(WIDTH,HEIGHT));
04          setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
05      }
06      public static void main(String[] args) {
07          JFrame jf=new JFrame();
08          jf.setVisible(true);
09
10      }
11      public static final int WIDTH=300;
12      public static final int HEIGH=200;
13
14  }
```

【运行结果】程序运行结果如图 11-8 所示。

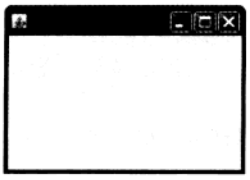


图 11-8　JFrame 效果

【代码解析】以上示例中，使用继承 JFrame 的方法创建了一个空白的窗体，设置窗体的可见属性为 true，最后在主方法中进行调用显示。

11.3.2　JDialog 窗体

JDialog 是创建对话框窗体的主要类，使用它主要是用来创建一个自定义的对话框。当然，也可以调用 JOptionPane 中的方法来创建标准的对话框。JDialog 中提供了一些构造方法，可以根据实际的要求来调用不同的构造方法。

- JDialog()：创建一个没有标题并且没有指定 Frame 所有者的无模式对话框。
- JDialog(Dialog owner)：创建一个没有标题但将指定的 Dialog 作为其所有者的无模式对

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

话框。

- JDialog(Dialog owner, boolean modal)：创建一个具有指定所有者 Dialog 和模式的对话框。
- JDialog(Dialog owner, String title)：创建一个具有指定标题和指定所有者的无模式对话框。
- JDialog(Dialog owner, String title, boolean modal) ：创建一个具有指定标题、模式和所有者 Dialog 的对话框。
- JDialog(Dialog owner, String title, boolean modal, GraphicsConfiguration gc)：创建一个具有指定标题、所有者 Dialog、模式和 GraphicsConfiguration 的对话框。
- JDialog(Frame owner)：创建一个没有标题但将指定的 Frame 作为其所有者的无模式对话框。
- JDialog(Frame owner, boolean modal)：创建一个具有指定所有者 Frame、模式和空标题的对话框。
- JDialog(Frame owner, String title)：创建一个具有指定标题和指定所有者 Frame 的无模式对话框。
- JDialog(Frame owner, String title, boolean modal)：创建一个具有指定标题、所有者 Frame 和模式的对话框。
- JDialog(Frame owner, String title, boolean modal, GraphicsConfiguration gc)：创建一个具有指定标题、所有者 Frame、模式和 GraphicsConfiguration 的对话框。
- JDialog(Window owner)：创建一个具有指定所有者 Window 和空标题的无模式对话框。
- JDialog(Window owner, Dialog.ModalityType modalityType)：创建一个具有指定所有者 Window、模式和空标题的对话框。
- JDialog(Window owner, String title)：创建一个具有指定标题和所有者 Window 的无模式对话框。
- JDialog(Window owner, String title, Dialog.ModalityType modalityType)：创建一个具有指定标题、所有者 Window 和模式的对话框。

在 JDialog 中也提供了一些比较常用的方法，如下所示。

- protected void addImpl(Component comp, Object constraints, int index)：添加指定的子组件。
- protected JRootPane createRootPane()：构造方法调用此方法来创建默认的根窗格。
- protected void dialogInit()：构造方法调用此方法来正确初始化 JDialog。
- AccessibleContext getAccessibleContext()：获取与此 JDialog 关联的 AccessibleContext。
- Container getContentPane()：返回此对话框的 contentPane 对象。
- int getDefaultCloseOperation()：返回用户在此对话框上启动 "close" 时所执行的操作。
- Component getGlassPane()：返回此对话框的 glassPane 对象。
- Graphics getGraphics()：为组件创建一个图形上下文。
- JMenuBar getJMenuBar()：返回此对话框上设置的菜单栏。
- JLayeredPane getLayeredPane()：返回此对话框的 layeredPane 对象。
- JRootPane getRootPane()：返回此对话框的 rootPane 对象。
- TransferHandler getTransferHandler()：获取 transferHandler 属性。
- static boolean isDefaultLookAndFeelDecorated()：如果应该通过当前外观为新创建的 JDialog 提供其 Window 装饰，则返回 true。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

- protected boolean isRootPaneCheckingEnabled()：返回是否将对 add 和 setLayout 的调用转发到 contentPane。
- protected String paramString()：返回此 JDialog 的字符串表示形式。
- protected void processWindowEvent(WindowEvent e)：根据 defaultCloseOperation 属性的状态处理窗体事件。
- void remove(Component comp)：从该容器中移除指定组件。
- void repaint(long time, int x, int y, int width, int height)：在 time 毫秒内重绘此组件的指定矩形区域。
- void setContentPane(Container contentPane)：设置 contentPane 属性。
- void setDefaultCloseOperation(int operation)：设置当用户在此对话框上启动 "close" 时默认执行的操作。
- void setGlassPane(Component glassPane)：设置 glassPane 属性。
- void setJMenuBar(JMenuBar menu)：设置此对话框的菜单栏。
- void setLayeredPane(JLayeredPane layeredPane)：设置 layeredPane 属性。
- void setLayout(LayoutManager manager)：设置 LayoutManager。
- protected void setRootPane(JRootPane root)：设置 rootPane 属性。
- protected void setRootPaneCheckingEnabled(boolean enabled)：设置是否将对 add 和 setLayout 的调用转发到 contentPane。
- void setTransferHandler(TransferHandler newHandler)：设置 transferHandler 属性，该属性是支持向此组件传输数据的机制。
- void update(Graphics g)：调用 paint(g)。

11.4 Swing 常用面板

JPanel 和前面所介绍的 Panel 类似。在 Swing 中，通常是将一些图画或者是文字信息描绘在 JPanel 上，然后再将 JPanel 示例放置到内容面板中。这时，可以通过布局管理器将组件合理放置到 JPanel 中，然后再使用布局管理器将 JPanel 放置到内容面板中。

JPanel 的父类是 JComponent，而 JComponent 继承了 Container 类，所以 JPanel 和 JFrame 一样，也可以放置 Swing 组件，而且还可以在上面书写文字、绘画等。JPanel 中提供了几种构造方法，可以根据实际情况进行选择。

- JPanel() ：创建具有双缓冲和流布局的新 JPanel。
- JPanel(boolean isDoubleBuffered)：创建是否使用双缓冲的 JPanel。
- JPanel(LayoutManager layout)：创建具有指定布局管理器的 JPanel。
- JPanel(LayoutManager layout, boolean isDoubleBuffered)：创建具有指定布局管理器和缓冲策略的新 JPanel。

JPanel 中也提供了一些比较常用的方法，如下所示。

- AccessibleContext getAccessibleContext()：获取与此 JPanel 关联的 AccessibleContext。
- PanelUI getUI()：返回呈现此组件的外观（L&F）对象。
- String getUIClassID()：返回指定呈现此组件的 L&F 类名的字符串。
- protected String paramString()：返回此 JPanel 的字符串表示形式。
- void setUI(PanelUI ui)：设置呈现此组件的外观（L&F）对象。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



- void updateUI()：利用当前外观的值重置 UI 属性。

【范例 11-9】示例代码 11-9 是一个测试面板的程序。

示例代码 11-9

```
01  public class JPanelTest
02  {
03      public static void main(String[] args) //主方法
04      {
05          PanelFrame frame=new PanelFrame(); //创建窗体对象
06          frame.addWindowListener(new        //给窗体对象添加关闭事件
07                  WindowAdapter(){
08              public void windowClosing(WindowEvent e)
09              {
10                  System.exit(0);
11              }
12          });
13          frame.setVisible(true);            //显示窗体
14      }
15  }
16  class PanelFrame extends JFrame            //继承 JFrame 类
17  {
18      public PanelFrame()
19      {
20          setTitle("使用面板");              //设置标题
21          setSize(400,300);                  //设置窗体大小
22          MyPanel p=new MyPanel();           //创建窗体面板
23          Container con=getContentPane();    //取得窗体容器
24          con.add(p);                        //向窗体添加面板
25      }
26  }
27  class MyPanel extends JPanel               //面板类继承自 JPanel 类
28  {
29      //在面板上显示信息
30      public void paintComponent(Graphics g) //重写 paintComponent 方法
31      {
32          g.drawString("这是显示在面板上的信息",120, 100);//显示信息
33      }
34  }
```

【运行结果】程序运行结果如图 11-9 所示。

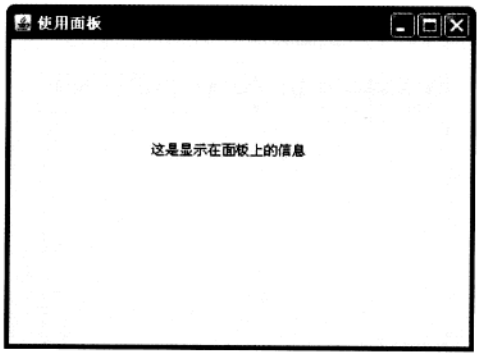


图 11-9　面板使用效果

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

11.5 习题

【习题 11-1】编写一个用户注册的界面程序，在该界面中要具有姓名、性别和年龄等内容。

【核心代码】

```
public class Register {
   public static void main(String[] args) {
      JFrame f = new JFrame("Register");
      Container c = f.getContentPane();
      c.setLayout(new BorderLayout());
      JPanel p = new JPanel();
      p.setLayout(new GridLayout(4, 2));
      p.setBorder(BorderFactory.createTitledBorder("请输入注册信息"));

      JLabel l1 = new JLabel("姓名：", JLabel.RIGHT);
      JLabel l2 = new JLabel("性别：", JLabel.RIGHT);
      JLabel l3 = new JLabel("年龄：", JLabel.RIGHT);
      JTextField t1 = new JTextField(10);
      JTextField t2 = new JTextField(10);
      JTextField t3 = new JTextField(10);
      p.add(l1);
      p.add(t1);
      p.add(l2);
      p.add(t2);
      p.add(l3);
      p.add(t3);

      JButton b = new JButton("提交");
      p.add(new Panel());
      p.add(b);

      c.add(p);
      f.setSize(500, 300);
      f.setVisible(true);
      f.addWindowListener(new WindowAdapter() {
         public void windowClosing(WindowEvent e) {
            System.exit(0);
         }
      });
   }
}
```

【习题 11-2】编写一个简单的文本编辑器，其中包括有菜单，如图 11-10 所示。

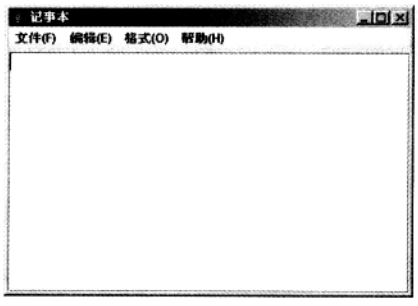


图 11-10 文本编辑器效果

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

【核心代码】

```
public class MenuBar {
    public static void main(String[] args) {
        JFrame f = new JFrame();
        Container c = f.getContentPane();
        c.setLayout(new BorderLayout());

        JMenuBar m = new JMenuBar();
        JMenu file = new JMenu("文件");
        JMenuItem newFile = new JMenuItem("新建");
        JMenuItem open = new JMenuItem("打开");
        JMenuItem close = new JMenuItem("关闭");
        JMenuItem exit = new JMenuItem("退出");
        file.add(newFile);
        file.add(open);
        file.add(close);
        file.addSeparator();
        file.add(exit);
        m.add(file);
        m.add(new JMenu("编辑"));
        m.add(new JMenu("格式"));
        m.add(new JMenu("帮助"));

        f.setJMenuBar(m);


        JTextArea t = new JTextArea();
        t.setEditable(true);

        c.add(new JScrollPane(t), BorderLayout.CENTER);

        f.setSize(500, 200);
        f.setVisible(true);
        f.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                System.exit(0);
            }
        });
    }
}
```

【习题 11-3】完善电话簿功能。

【核心代码】

```
public class PhoneBook {
    public static void main(String[] args) {
        JFrame f = new JFrame();
        Container c = f.getContentPane();
        Box box, box1, box2;
        JTextField name, email, phone;
        JButton button;

        name = new JTextField(12);
        email = new JTextField(12);
        phone = new JTextField(12);
        button = new JButton("录入");

        box1 = Box.createVerticalBox();
        box1.add(new JLabel("输入姓名"));
        box1.add(Box.createVerticalStrut(8));
        box1.add(new Label("输入 email"));
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
        box1.add(Box.createVerticalStrut(8));
        box1.add(new Label("输入电话"));
        box1.add(Box.createVerticalStrut(8));
        box1.add(new Label("单击录入"));

        box2 = Box.createVerticalBox();
        box2.add(name);
        box2.add(Box.createVerticalStrut(8));
        box2.add(email);
        box2.add(Box.createVerticalStrut(8));
        box2.add(phone);
        box2.add(Box.createVerticalStrut(8));
        box2.add(button);

        box = Box.createHorizontalBox();
        box.add(box1);
        box.add(Box.createHorizontalStrut(10));
        box.add(box2);

        c.add(box);

        f.setSize(500, 300);
        f.setVisible(true);
        f.addWindowListener(new WindowAdapter() {
           public void windowClosing(WindowEvent e) {
              System.exit(0);
           }
        });
     }
}
```

【习题 11-4】创建一个窗口，在窗口中设置两个文本框，将用户在第一个文本框中输入的内容显示到第二个文本框中。

【核心代码】

```
public class InputText {
   public static void main(String[] args) {
      JFrame f = new JFrame("Register");
      Container c = f.getContentPane();
      c.setLayout(new BorderLayout());
      Box box, box1, box2;
      final JTextArea t1, t2;
      t1 = new JTextArea();
      t2 = new JTextArea();

      JButton b = new JButton("提交");
      b.addActionListener(new ActionListener() {

         public void actionPerformed(ActionEvent e) {
            t1.setText(t1.getText() + t2.getText());
            t2.setText("");
         }
      });

      box1 = Box.createVerticalBox();
      box1.add(new JScrollPane(t1), BorderLayout.CENTER);

      box2 = Box.createVerticalBox();
      box2.add(new JScrollPane(t2), BorderLayout.CENTER);
      box2.add(Box.createVerticalStrut(3));
      box2.add(b);
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
        box = Box.createVerticalBox();
        box.add(box1);
        box.add(Box.createVerticalStrut(20));
        box.add(box2);

        c.add(box);

        f.setSize(500, 300);
        f.setVisible(true);
        f.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent
        e) {
                System.exit(0);
            }
        });
    }
}
```

【习题 11-5】创建一个窗口，在窗口中有两个区域，上方是文本框，下方是一组复选框，列出所有学生待选的课程，将所选择的课程显示在上面的文本框中。

【核心代码】

```
public class SelectCourse {
    public static void main(String[] args) {
        JFrame f = new JFrame();
        final Container c = f.getContentPane();
        Box box, box1, box2;
        JTextField name, email, phone;
        final JTextArea t1 = new JTextArea();
        JPanel p = new JPanel();
        p.setLayout(new GridLayout(2, 3));
        p.setBorder(BorderFactory.createTitledBorder("请选择课程"));
        final JCheckBox c1 = new JCheckBox("组织行为学");
        final JCheckBox c2 = new JCheckBox("管理经济学");
        final JCheckBox c3 = new JCheckBox("市场营销学");
        final JCheckBox c4 = new JCheckBox("财务会计学");
        final JCheckBox c5 = new JCheckBox("创业管理学");
        final JCheckBox c6 = new JCheckBox("企业管理学");
        p.add(c1);
        p.add(c2);
        p.add(c3);
        p.add(c4);
        p.add(c5);
        p.add(c6);
        JButton button = new JButton("提交");
        button.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                StringBuffer sb = new StringBuffer();
                if (c1.isSelected()) sb.append(c1.getText()).append("   ");
                if (c2.isSelected()) sb.append(c2.getText()).append("   ");
                if (c3.isSelected()) sb.append(c3.getText()).append("   ");
                if (c4.isSelected()) sb.append(c4.getText()).append("   ");
                if (c5.isSelected()) sb.append(c5.getText()).append("   ");
                if (c6.isSelected()) sb.append(c6.getText()).append("   ");
                t1.setText(sb.toString());
            }
        });

        box1 = Box.createVerticalBox();
        box1.add(new JScrollPane(t1), BorderLayout.CENTER);

        box2 = Box.createVerticalBox();
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
        box2.add(p, BorderLayout.CENTER);
        box2.add(Box.createVerticalStrut(3));
        box2.add(button);

        box = Box.createVerticalBox();
        box.add(box1);
        box.add(Box.createVerticalStrut(20));
        box.add(box2);

        c.add(box);

        f.setSize(500, 300);
        f.setVisible(true);
        f.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                System.exit(0);
            }
        });
    }
}
```

【习题 11-6】设计一个计算器，使它能够进行简单的计算。

【核心代码】

```
public class Calculator {
    public static void main(String[] args) {
        JFrame f = new JFrame();
        final Container c = f.getContentPane();
        JTextField result = new JTextField("0");
        result.setEditable(false);
        result.setHorizontalAlignment(JTextField.RIGHT);
        result.setColumns(12);
        JButton clear = new JButton("C");
        JButton point = new JButton(".");
        JButton count = new JButton("=");
        JButton add = new JButton("+");
        JButton subtract = new JButton("_");
        JButton multiply = new JButton("X");
        JButton divide = new JButton("/");
        JButton back = new JButton("Back");
        JButton percent = new JButton("%");
        JButton inverse = new JButton("1/x");

        JPanel p = new JPanel();
        p.setLayout(new BorderLayout());
        p.add(result, BorderLayout.NORTH);
        JPanel p1 = new JPanel();
        p1.setLayout(new GridLayout(4, 4));
        p1.add(clear);
        p1.add(back);
        p1.add(count);
        p1.add(point);
        p1.add(add);
        p1.add(subtract);
        p1.add(multiply);
        p1.add(divide);
        p1.add(percent);
        p1.add(inverse);

        for (int i = 0; i < 10; i++) {
            p1.add(new JButton("" + i));
        }
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
        p.add(p1);

        c.add(p);

        f.setSize(500, 300);
        f.setVisible(true);
        f.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                System.exit(0);
            }
        });
    }
}
```

【习题 11-7】创建一个窗口，在窗口中提供有一组单选按钮可以控制文本框中文字的字号。

【核心代码】

```
public class FontSize {
    public static void main(String[] args) {
        JFrame f = new JFrame();
        Container c = f.getContentPane();
        c.setLayout(new BorderLayout());

        JPanel p = new JPanel();
        p.setLayout(new BorderLayout());


        JPanel sp = new JPanel();
        JRadioButton r9 = new JRadioButton("9");
        JRadioButton r12 = new JRadioButton("12");
        JRadioButton r18 = new JRadioButton("18");
        JRadioButton r22 = new JRadioButton("22");
        sp.add(r9);
        sp.add(r12);
        sp.add(r18);
        sp.add(r22);
        ButtonGroup bg = new ButtonGroup();
        bg.add(r9);
        bg.add(r12);
        bg.add(r18);
        bg.add(r22);

        p.add(sp, BorderLayout.NORTH);

        final JTextArea t = new JTextArea();
        p.add(new JScrollPane(t), BorderLayout.CENTER);

        r9.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                t.setFont(new Font("TimeRoman", Font.PLAIN, 9));
            }
        });
        r12.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                t.setFont(new Font("TimeRoman", Font.PLAIN, 12));
            }
        });
        r18.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                t.setFont(new Font("TimeRoman", Font.PLAIN, 18));
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
            }
        });
        r22.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                t.setFont(new Font("TimeRoman", Font.PLAIN, 22));
            }
        });

        c.add(p);

        f.setSize(500, 200);
        f.setVisible(true);
        f.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                System.exit(0);
            }
        });
    }
}
```

【习题 11-8】创建一个窗口，使单击一个按钮，弹出一个确认或取消对话框。

【核心代码】

```
public class OptionDialog {
    public static void main(String[] args) {
        JFrame f = new JFrame("Register");
        Container c = f.getContentPane();

        final JButton b = new JButton("Popup Dialog");
        c.add(b);
        b.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                int ch = JOptionPane.showConfirmDialog(b, "请选择！");
                if (ch == 0) {
                    System.out.println("yes");
                } else if (ch == 1) {
                    System.out.println("no");
                } else if (ch == 2) {
                    System.out.println("cancel");
                }
            }
        });

        f.setSize(500, 300);
        f.setVisible(true);
        f.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent
        e) {
                System.exit(0);
            }
        });
    }
}
```

【习题 11-9】创建 Swing 菜单栏。

【核心代码】

```
JMenuBar m = new JMenuBar();
JMenu file = new JMenu("文件");
JMenuItem newFile = new JMenuItem("新建");
JMenuItem open = new JMenuItem("打开");
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
JMenuItem close = new JMenuItem("关闭");
JMenuItem exit = new JMenuItem("退出");
file.add(newFile);
file.add(open);
file.add(close);
file.addSeparator();
file.add(exit);
m.add(file);
m.add(new JMenu("编辑"));
m.add(new JMenu("格式"));
m.add(new JMenu("帮助"));

f.setJMenuBar(m);
```

【习题 11-10】创建一个监视任务的进度条。

【核心代码】

```
public class ProgressBar {
   public static void main(String[] args) {
      JFrame f = new JFrame();
      Container c = f.getContentPane();
      c.setLayout(new BorderLayout());

      final ProgressMonitor monitor = new ProgressMonitor(f, "Monitoring the
       progress", "", 0, 100);
      monitor.setProgress(0);

      JButton b = new JButton("Click to monitor the progress.");
      c.add(b);

      b.addActionListener(new ActionListener(){
         public void actionPerformed(ActionEvent e) {
            ProgressTask task = new ProgressTask(monitor);
            task.start();
         }
      });

      f.setSize(500, 200);
      f.setVisible(true);
      f.addWindowListener(new WindowAdapter() {
         public void windowClosing(WindowEvent e) {
            System.exit(0);
         }
      });
   }

   static class ProgressTask implements Runnable {
      private boolean running;
      private Thread t;
      private ProgressMonitor monitor;
      private int value;


      public ProgressTask(ProgressMonitor monitor) {
         this.monitor = monitor;
         running = false;
      }

      public void run() {
         while (running) {
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
                try {
                    Thread.sleep(200);
                    if (monitor.isCanceled()) {
                        running = false;
                    }
                    monitor.setProgress(++value);
                    monitor.setNote(value + "% complete");
                    if (value >= 100) {
                        running = false;
                    }
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }

            }
        }

        public void start() {
            if (! running) {
                value = 0;
                monitor.setProgress(value);
                running = true;
                t = new Thread(this);
                t.start();
            }
        }

        public void stop() {
            monitor.setProgress(0);
            t = null;
        }

    }
}
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第 12 章　事 件 处 理

事件的处理是图形界面和用户之间进行交互的重要内容。事件的处理包括有事件源、事件和事件处理器三个概念。一个事件的处理首先是要为事件源注册相应的事件处理器，事件源产生指定事件，事件处理器捕获之后就要进行响应的处理了。本章中主要介绍事件处理的相关内容。



12.1　事件类

每一个事件类都与一个事件类接口相对应，由事件引起的动作都存放在接口需要实现的方法中。本节主要讲述 Java 中比较常用的事件类接口，即动作事件类、调整事件类、焦点事件类、项目事件类、按钮事件类、鼠标事件类及窗口事件类。

12.1.1　事件类分类

所有与 AWT 相关的事件类都是 java.awt.AWTEvent 的派生类，AWTEvent 又是 java.util.EventObject 类的派生类。事件类的派生关系如下所示。

```
java.lang.Object
  +--java.util.EventObject
     +--java.awt.AWTEvent
       +--java.awt.event.*Event
```

总体来说，AWT 事件有低级事件和高级事件两大类。低级事件是指源于组件或容器的事件，当组件或容器发生事件时（鼠标单击左键、右键，鼠标拖动，以及改变窗口大小等），将触发事件。高级事件是语义事件，此类事件与特定的具体事件不一定相对应，但是会产生特定的事件对象，如按钮被按下触发 ActionEvent 事件、滚动条移动滑块触发 AdjustmentEvent 事件、选中项目列表某项时触发 ItemEvent 事件等。低级事件包括以下几种。

- 组件事件（ComponentEvent）：组件被移动、大小被改变或可见性被修改的低级事件。
- 容器事件（ContainerEvent）：容器内容因添加、删除组件的改变而产生的事件。
- 窗口事件（WindowEvent）：主要用于对窗口大小进行缩放等操作的事件。
- 焦点事件（FocusEvent）：组件获得焦点或者失去焦点的事件。
- 键盘事件（KeyEvent）：获取键盘操作的事件。
- 鼠标事件（MouseEvent）：鼠标单击、双击或者是移动的事件操作。

高级事件（语义事件）包括以下几种。

- 动作事件(ActionEvent)：组件定义的动作事件。
- 调整事件(AdjustmentEvent)：由实现 Adjustable 接口的组件发出的事件。
- 项目事件(ItemEvent)：项目被选中或取消选中所发出的事件
- 文本事件（TextEvent）：指对象文本改变的语义事件。

下面将介绍几个经常使用的事件，其他的事件也十分类似，如果遇到什么问题，读者可以查询 API 等相关文档。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

12.1.2 动作事件类

动作事件类就是用户在操作按钮、多选框、文本框等组件的时候所发生的动作事件，也就是指发生组件所定义的语义事件，如单击按钮、在文本框中按下回车键等。使用动作事件类的时候需要为组件添加一个事件监听器 ActionListener，在该监听器中只存在有一个方法 actionPerfomed()方法，该方法的基本形式如下。

```
Public void actionPerformed(ActionEvent e){
   //按钮被操作发生
}
```

假设存在按钮组件对象 button，动作事件就应该写成如下的形式。

```
button.addActionListener(new ActionListener()
{
  public void actionPerformed(ActionEvent e)
  {
    //按钮被操作
  }
}
```

在上面的程序中，定义了一个事件监听类，并且实现了 ActionListener 接口，需要在按钮 button 中通过 addActionListener()方法进行事件监听器的注册。

12.1.3 调整事件类

调整事件类主要是为滑动滚动条的滚动块所设定的事件。滚动条组件 ScrollBar 在滑动滚动块的时候就会触发这个事件。和前面相同，如果要监听调整事件，就需要为 ScrollBar 添加一个调整事件监听器 AdjustmentListener。在这个监听器中也只有一个方法 adjustmentValueChanged()，只需要实现该方法就可以了，该方法的基本形式如下。

```
public void adjustmentValueChanged(AdjustmentEvent e)
{
  //调整滚动条发生
}
```

假设现有滚动条组件对象 Scrollbar，调整事件使用方法如下。

```
scrollbar.addAdjustmentListener(new AdjustmentListener()
{
  public void adjustmentValueChanged(AdjustmentEvent e)
  {
    //调整滚动条 do something
  }
});
```

12.1.4 焦点事件类

焦点事件类指的是用户程序界面中的组件失去焦点或者是获得焦点时所发生的事件。例如，现在正在使用铅笔，那么就相当于铅笔获得了焦点，一会我再换个钢笔，就是铅笔失去了焦点，而钢笔获得了焦点。所以，得到焦点的组件就会处于一个被激活的状态，平时在注册论坛用户时，可以看到光标在文本输入框中闪动，这就说明当前的文本框是拥有焦点的，向文本框中输入文字时，就会触发焦点事件。使用焦点事件就需要给组件增加一个 FocusListener 接口的事件处理器，在这个接口中共有两个方法，如下所示。

```
void focusGained(FocusEvent e)
{
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
 //当获得焦点时发生
}
void focusLost(FocusEvent e)
{
//当失去焦点时发生
}
```

一个窗口中的组件在同一时间，只能够有一个组件获得焦点。如果单击了另一个组件，此时该组件就会失去焦点。在 Java 中与焦点相关的方法如下。

- FocusOwner：焦点拥有者，也就是持有焦点的组件。
- FocusWindow：焦点窗口，含有焦点拥有者的窗口。
- ActiveWindow：活动窗口，包含有焦点的框架或者对话框。

12.1.5　项目事件类

项目事件类（ItemEvent）是为制定一个项目被选定或者是被取消的事件，例如，选择一个单选按钮、复选框或者是列表等组件时，就会产生一个项目事件。要使用这个事件，就需要为组件添加一个实现 ItemListener 接口的事件处理器。这个接口比较常用的主要方法如下。

- getItem()：返回取得影响的项目对象。
- getItemSelectable()：返回事件源 ItemSelectable 对象。
- getStateChange()：返回状态的改变类型，包括 SELECTED 和 DESELECTED 两种。

12.1.6　键盘事件类

键盘事件类（KeyEvent）是容器内的任意一个组件获得焦点的时候，组件所发生的按下键盘的事件。如果按下键盘上的某一个按键或者是某一组合按键，就会触发该键盘事件，例如，在操作系统中按下“Alt+F4”组合键，就会关闭当前的窗口。要使用键盘事件，就需要为组件添加一个 KeyListener 接口的事件处理器。在这个接口中主要包括有三个方法，如下所示。

```
void keyPressed(KeyEvent e)
{
// 按下按键时的事件
}
void keyReleased(KeyEvent e)
{
//放开按键时的事件
}
void keyTyped(KeyEvent e)
{
//敲击键盘，发生在按键按下后，按键放开前
}
```

键盘事件类的一些方法如下所示。

- getKeyChar()：返回在键盘上按下的字符。
- getKeyCode()：返回在键盘上按下的字符码。
- getKeyLocation()：返回键位置。
- getKeyModifiersText()：返回描述修饰符的文本字符串。
- getKeyText()：返回键的 keyCode 标识字符串。
- isActionKey()：判断键是否是操作键。
- setKeyChar()：改变键字符为指定的字符。
- setModifiers(int modifiers)：改变键修饰符为指定的键修饰符。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

- paramString()：生成事件状态的字符串。

【范例 12-1】示例代码 12-1 是一个测试键盘事件类的程序。

示例代码 12-1

```
01  public class KeyTest extends KeyAdapter
02          implements ActionListener {
03      JFrame jframe = null;           //声明一个 JFrame 类对象
04      JLabel label = null;            //声明一个 JLabel 类对象
05      JTextField jtext = null;        //声明一个 JTextField 类对象
06      String key = "";
07      public KeyTest() {
08          jframe = new JFrame("键盘监听示例"); // 为这个 JFrame 设置一个标题
09          Container contentPane = jframe.getContentPane();
10          contentPane.setLayout(new GridLayout(3, 1));
11          label = new JLabel();               // 创建一个标签对象
12          jtext = new JTextField();
13          jtext.requestFocus();
14          jtext.addKeyListener(this);
15          JButton b = new JButton("清除");    // 创建一个按钮
16          b.addActionListener(this);
17          contentPane.add(label);
18          contentPane.add(jtext);
19          contentPane.add(b);
20          jframe.pack();
21          jframe.show();
22          jframe.addWindowListener(new WindowAdapter() {  //添加窗口事件监听
23              public void windowClosing(WindowEvent e) {
24                  System.exit(0);
25              }
26          });
27      }
28      public void actionPerformed(ActionEvent e) {
29          key = "";
30          label.setText("");
31          jtext.setText("");
32          jtext.requestFocus();
33      }
34      public void keyTyped(KeyEvent e) {
35          char c = e.getKeyChar();  //获取事件源上的字符
36          if (c == 'o') {           //如果输入字母 o，就会产生新窗口
37              JFrame newF = new JFrame("这是输入 o 后弹出的新窗口");
38              newF.setSize(200, 200);
39              newF.show();
40          }
41          key = key + c;
42          label.setText(key);
43      }
44      public static void main(String[] args) {
45          new KeyTest();
46      }
47  }
```

【运行结果】程序运行结果如图 12-1 所示。

【代码解析】以上程序中，声明窗口对象，并在窗口中加入文本框、标签、按钮组件。然后为按钮设置事件监听，如果单击按钮，则清除文本框和标签中的文字。为文本框设置事件监听，如果输入“o”，则打开一个新的窗口。

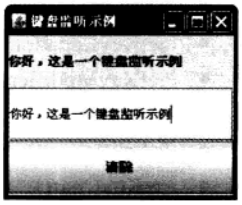


图 12-1　键盘事件效果

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



12.1.7　鼠标事件类

鼠标事件类（MouseEvent）指组件中发生的鼠标动作事件，如按下鼠标、释放鼠标、单击鼠标、鼠标光标进入或离开组件的几何图形、移动鼠标、拖动鼠标等。当鼠标移动到某个区域或鼠标单击某个组件时，就会触发鼠标事件。要使用鼠标事件，必须给组件添加一个 MouseListener 接口的事件处理器，该接口包含五个方法，如下所示。

```
void mouseClicked(MouseEvent e)
{
//当鼠标在该区域单击时发生
}
void mouseEntered(MouseEvent e)
{
//当鼠标进入该区域时发生
}
void mouseExited(MouseEvent e)
{
//当鼠标离开该区域时发生
}
void mousePressed(MouseEvent e)
{
//当鼠标在该区域按下时发生
}
void mouseReleased(MouseEvent e)
{
//当鼠标在该区域放开时发生
}
```

鼠标事件类的方法如下所示。

- getButton()：返回鼠标按键状态改变指示。
- getClickCount()：返回鼠标按键单击的次数。
- getMouseModifiersText()：返回指定修饰符文本字符串。
- getPoint()：返回事件源中位置对象。
- getX()：返回鼠标在指定区域内相对位置的横坐标。
- getY()：返回鼠标在指定区域内相对位置的纵坐标。
- paramString()：生成事件状态的字符串。

【范例 12-2】示例代码 12-2 是一个测试鼠标事件类的程序。

示例代码 12-2

```
01  public class MouseTest extends WindowAdapter
02                implements MouseListener {
03      JFrame jfarme = null;
04      JButton button1 = null;
05      JLabel label = null;
06      public MouseTest() {
07          jfarme = new JFrame("鼠标事件示例");                    //创建一个框架
08          Container contentPane = jfarme.getContentPane(); //获得一个窗口容器
09          contentPane.setLayout(new GridLayout(2, 1));      //设置布局方式
10          button1 = new JButton("按钮");                      //创建一个按扭
11  label = new JLabel("起始状态，还没有鼠标事件", JLabel.CENTER);//创建一个文本标签
12          button1.addMouseListener(this);              //为按扭添加事件监听
13          contentPane.add(label);
14          contentPane.add(button1);
15          jfarme.pack();                                      //将窗口调节到适当的大小
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
16            jfarme.show();                                    //显示窗口
17            jfarme.addWindowListener(this);                   //添加窗口事件监听
18        }
19        public void mousePressed(MouseEvent e) {              //激发按下鼠标事件
20            label.setText("你已经按下鼠标按钮");
21        }
22        public void mouseReleased(MouseEvent e) {             //激发释放鼠标按键
23            label.setText("你已经抬起鼠标按钮");
24        }
25        public void mouseEntered(MouseEvent e) {              //鼠标进入组件时调用
26            label.setText("光标已经进入按钮");
27        }
28        public void mouseExited(MouseEvent e) {               //鼠标离开组件时调用
29            label.setText("光标已经离开按钮");
30        }
31        public void mouseClicked(MouseEvent e) {              //鼠标在组件上单击时调用
32            label.setText("你单击按钮");
33        }
34        public void windowClosing(WindowEvent e) {
35            System.exit(0);
36        }
37        public static void main(String[] args) {
38            new MouseTest();
39        }
40    }
```

【运行结果】程序运行结果如图 12-2～图 12-4 所示。



图 12-2　光标离开按钮



图 12-3　按下按钮



图 12-4　单击按钮

【代码解析】MouseMotionListener 主要负责鼠标的拖动事件处理。当用户在添加了 MouseMotionListener 监听器的组件上拖动鼠标时，就会激活 mouseDragged(MouseEvent e)方法，如果用户此时没有按下鼠标而是移动鼠标时，就会激活 mouseMoved(MouseEvent e)方法。

12.1.8　窗口事件类

窗口事件（WindowEvent）指窗口状态改变的事件，如窗口 Window 对象的打开 、关闭、激活、停用，或者焦点转移到窗口内，以及焦点移除而生成的事件。一般发生在 Window、Frame、Dialog 等类的对象上。使用窗口事件必须为组件添加一个实现 WindowListener 接口的事件处理器，该接口包含七个方法，如下所示。

```
void windowActivated(WindowEvent e)
{
// 窗口被激活时发生
}
void windowClosed(WindowEvent e)
{
// 窗口关闭之后发生
}
void windowClosing(WindowEvent e)
{
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
// 窗口关闭过程中发生
}
void windowDeactivated(WindowEvent e)
{
// 窗口不再处于激活状态时发生
}
void windowDeiconified(WindowEvent e)
{
// 窗口大小从最小到正常时发生
}
void windowIconified(WindowEvent e)
{
// 窗口从正常到最小时发生
}
void windowOpened(WindowEvent e)
{
//窗口第一次被打开时所发生
}
```

在窗口事件类中有以下几种方法。

- getNewState()：返回窗口改变之后的新状态。
- getOldState()：返回窗口改变之后的旧状态。
- getOppositeWindow()：返回事件涉及的辅助窗口。
- getWindow()：返回事件源。
- paramString()：生成事件状态的字符串。



12.2　事件监听器

事件监听器是用于监听事件源触发事件的对象，其中包括有对事件发生后的处理操作。在 Java 语言中，针对不同的事件，定义了相应的事件监听接口。下面是比较常用的事件监听器接口。

- ActionListener：用于接收操作事件的监听器接口。
- AdjustmentListener：用于接收调整事件的监听器接口。
- AWTEventListener：用于接收指派给对象的事件的通知。
- ComponentListener：用于接收组件事件的监听接口。
- ContainerListener：用于接收容器事件的监听接口。
- FocusListener：用于接收组件上的键盘焦点事件的监听接口。
- HierarchyBoundsListener：用于接收组件上的移动和大小调整事件的监听接口。
- InputMethodListener：接收输入方法事件的监听接口。
- KeyListener：用于接收键盘事件的监听接口。
- MouseListener：接收组件上的鼠标事件（包括按下、单击、进入或者离开）的监听器接口。
- MouseMotionListener：接收组件上的鼠标移动事件的监听接口。
- MouseWheelListener：接收组件上的鼠标滚轮事件的监听接口。
- TextListener：接收文本事件的监听器接口。
- WindowListener：接收窗口事件的监听接口。

使用监听器，首先要定义监听器类，并实现相应的监听器接口，然后再在组件上使用 addXxxxListener 的方式为组件添加事件监听，最后设置相应的事件处理方法。当组件中的事件

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

触发后，就会根据所添加的事件处理方法进行事件处理。

12.2.1 事件监听器的接口实现

前面所介绍的每一个事件类都有一个事件监听器的接口，例如，实现键盘事件就必须要实现接口 EventListener，重写接口的方法如下。

```
public interface KeyListener extends EventListener
{
    public void keyPressed(KeyEvent ev);
    public void keyReleased(KeyEvent ev);
    public void keyTyped(KeyEvent ev);
}
```

从以上的代码中可以看出，键盘事件接口中有三个方法，分别是 keyPressed()、keyReleased()、keyTyped()。当按下键盘的某一个按键时，会调用 keyPressed()方法；松开按键时，就会调用 keyReleased()方法；而在按键从按下到抬起的这段时间，就会调用 keyTyped()方法。

12.2.2 应用事件监听器

下面是一个事件监听器的使用示例。

【范例 12-3】示例代码 12-3 是一个鼠标事件类程序。

示例代码 12-3

```
01  public class EventTest1 extends WindowAdapter implements ActionListener {
02      JButton b1 = null;
03      JButton b2 = null;
04      public EventTest1() {                                   //构造方法
05          JFrame f = new JFrame("EventTest1");                //创建一个框架
06      Container contentPane = f.getContentPane();             //获得一个窗口容器
07      contentPane.setLayout(new GridLayout(1, 2));            //设置布局方式
08      b1 = new JButton("关闭窗口");                            //创建一个关闭窗口的按扭
09      b2 = new JButton("打开窗口");                            //创建一个打开新窗口的按扭
10      //为按扭添加事件监听
11      b1.addActionListener(this);
12      b2.addActionListener(this);
13      //往容器中添加按扭组件
14      contentPane.add(b1);
15      contentPane.add(b2);
16      f.pack();                                   //将窗口调节到适当的大小
17      f.show();                                   //显示窗口
18      f.addWindowListener(this);                  //添加窗口事件监听
19  }
20  public void actionPerformed(ActionEvent e) {
21      if (e.getSource() == b1)                    // 利用 getSource 判断哪个按钮被按下了。
22          Toolkit.getDefaultToolkit().beep();
23      if (e.getSource() == b2) {
24          JFrame jf = new JFrame("新窗口");
25          jf.setSize(200, 200);
26          jf.show();
27      }
28  }
29  public void windowClosing(WindowEvent e) {
30      System.exit(0);
31  }
32  public static void main(String args[]) {
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
33          new EventTest1();
34      }
35  }
```

【运行结果】程序运行结果如图 12-5 所示。



图 12-5　应用事件监听器效果

【代码解析】以上程序中，声明窗口对象，并在窗口中设置两个按钮。然后为两个按钮分别设置事件监听，当单击第一个按钮时，关闭窗口；当单击第二个按钮时，则打开一个新的窗口。最后为窗口设置显示属性为 true。

12.2.3　事件监听器的特点

一个监听器接口的类可以实现一个或者多个接口，接口之间可以使用逗号隔开，一个组件可以监听一个或者是多个事件源的多个事件，使用方法如下。

```
button.addMouseMotinListener(this);          //添加鼠标动作监听
button.addMouseListener(this);               //添加鼠标监听器
button b.addWindowListener(this);            //添加窗口监听器
```

上面的语法代码就是在对象 button 上接收多个事件，并且是被同一个监听器接收并进行处理。

事件的处理者和事件源可以同处于一个类中。

12.3　事件适配器

前面一节中介绍了事件的监听器，都是以实现事件监听器的接口方式进行定义的。如果要实现接口，必须实现接口中的所有方法。在 Java 中定义了每个事件接听接口相应的实现类，这就是事件适配器。用户可以继承使用事件适配器，这样只需要覆盖一些必要的事件处理方法就可以了，会使得代码变得更加简洁，同时，也不必像使用接口那样实现全部的方法。下面是比较常用的事件适配器。

- ComponentAdapter：接收组件事件的抽象适配器。
- ContainerAdapter：接收容器时间的抽象适配器。
- FocusAdapter：接收键盘焦点事件的抽象适配器。
- KeyAdapter：接收键盘事件的抽象适配器。
- MouseAdapter：接收鼠标事件的抽象适配器类。
- MouseMotionAdapter：接收鼠标移动事件的抽象适配器类。
- WindowAdapter：接收窗口事件的抽象适配器类。

【范例 12-4】示例代码 12-4 是一个窗口事件类程序。

示例代码 12-4

```
01  public class WindowAdapterTest {
02      public static void main(String[] args) {
03
04          Frame fra=new Frame("窗口关闭示例");
05          fra.setSize(200,200);
06          fra.setVisible(true);
07          fra.addWindowListener(new WindowClose());
08      }
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
09  }
10  class WindowClose extends WindowAdapter{
11      public void windowClosing(WindowEvent e)
12      {
13          System.exit(1);
14
15      }
16  }
```



图 12-6　事件适配器使用效果

【运行结果】程序运行结果如图 12-6 所示。

【代码解析】在以上程序中，首先创建一个窗口，然后设置窗口关闭的事件，当单击窗口中的“”按钮时，就会关闭窗口。

12.4　事件应用方法

在使用事件适配器或者是事件监听器的时候，可以根据实际的情况而选择使用内部类或者是外部类的方式。本节主要介绍使用内部类、外部类、匿名内部类在图形界面开发中的应用。

12.4.1　内部类方式应用

前面介绍过内部类，也就是定义在一个类内部的类，它的特点就是对象具有外部类变量和方法的访问权限包括有私有成员，而且使用内部类，在实现事件监听器的功能时比较方便简洁。

【范例 12-5】示例代码 12-5 是一个测试内部类方式的程序。

示例代码 12-5

```
01  public class MouseInnerTest {
02      private Frame f;
03      private TextField tf;
04      public static void main(String[] args) {
05
06          MouseInnerTest mit=new MouseInnerTest();
07          mit.show();
08
09      }
10      public MouseInnerTest(){
11          f=new Frame("监听器");
12          tf=new TextField();
13      }
14      public void show()
15      {
16          f.add(tf,"North");
17          f.setVisible(true);
18          f.addMouseMotionListener(new MyMouseMotionListener());
19          f.setSize(300,240);
20          f.addWindowListener(new WindowAdapter(){
21              public void windowClosing(WindowEvent e) {
22                  f.setVisible(false);
23                  f.dispose();
24                  System.out.println("退出");
25                  System.exit(0);
26              }
27          });
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
28        }
29        class MyMouseMotionListener extends MouseMotionAdapter{
30            public void mouseMoved(MouseEvent e){
31                tf.setText("鼠标坐标：X="+e.getX()+"Y="+e.getY());
32            }
33        }
34    }
```

【运行结果】程序运行结果如图 12-7 所示。

【代码解析】在以上程序中，创建了一个窗口对象，在窗口中添加了一个文本框组件，并添加了一个鼠标事件监听，获取到鼠标在窗口中的坐标位置，并在文本框中进行显示。



图 12-7　内部类方式效果图

12.4.2　匿名类方式应用

事件处理中，不仅可以使用成员内部类，也可以使用匿名内部类的方式。声明了成员内部类以后，只创建一个对象，并且新的类继承了事件适配器或者是实现了事件的监听接口，就可以使用匿名内部类。

【范例 12-6】示例代码 12-6 是一个测试匿名类方式的类程序。

示例代码 12-6

```
01  public class JButtonTest {
02      public JButtonTest() {
03  JFrame f = new JFrame("打开窗口");//创建一个 JFrame 对象，并设置它的标题为"打开窗口"
04          Container contentPane = f.getContentPane();
05          contentPane.setLayout(new GridLayout(1, 2));
06          JButton b1 = new JButton("打开一个文本窗口"); //定义一个按钮组件
07          JButton b2 = new JButton("打开一个空窗口");
08          b1.setMnemonic('T');                    //设置 b1 上的键盘助记符为 T
09          b2.setMnemonic('I');                    //设置 b2 上的键盘助记符为 I
10          f.getRootPane().setDefaultButton(b1);
11          b1.addActionListener(                   //添加监听器
12                  new ActionListener() {
13                      public void actionPerformed(ActionEvent e) {
14                          JFrame newf = new JFrame("文字窗口");
15                          JLabel label = new JLabel("这是一个有文字的窗口");
16                          label.setHorizontalAlignment(JLabel.CENTER);
17                          newf.getContentPane().add(label);
18                          newf.setSize(200, 200);
19                          newf.show();
20                      }
21                  });
22          b2.addActionListener(                   //添加监听器
23                  new ActionListener() {
24                      public void actionPerformed(ActionEvent e) {
25                      JFrame newf = new JFrame("空窗口");//创建一个 JFrame 的对象
26                          JLabel label = new JLabel();
27                          label.setHorizontalAlignment(JLabel.CENTER);
28                          newf.getContentPane().add(label);
29                          newf.setSize(200, 200);      //设置窗口的大小
30                          newf.show();
31                      }
32                  });
33          //将这两个按钮添加到面板中
34          contentPane.add(b1);
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
35            contentPane.add(b2);
36            f.pack();
37            f.show();
38            f.addWindowListener(new WindowAdapter() {
39                public void windowClosing(WindowEvent e) {
40                    System.exit(0);
41                }
42            });
43        }
44        public static void main(String[] args) {
45            new JButtonTest();
46        }
47    }
```

【运行结果】程序运行结果如图 12-8 所示。

【代码解析】在以上程序中，创建了一个窗口对象，并在窗口中设置了两个按钮。为两个按钮添加事件监听，并使用匿名内部类的方式进行处理，当单击第一个按钮时，会弹出一个带有文字的窗口；单击第二个按钮时，会弹出一个空的窗口。

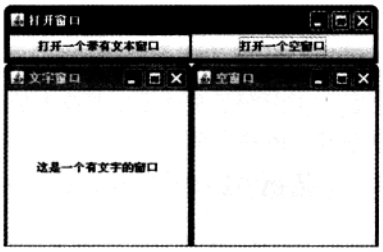


图 12-8　匿名内部类运行效果

12.4.3　外部类方式应用

事件的应用处理方法也可以使用外部类的方式。事件监听和事件处理类分开进行管理。下面的示例是将前面显示鼠标坐标的示例进行修改，将监听器类单独分开。

【范例 12-7】示例代码 12-7 是一个外部类写法示例。

示例代码 12-7

```
01  public class MyMouseTest {
02      private Frame f;
03      private TextField tf;
04      public static void main(String[] args) {
05
06          MouseInnerTest mit=new MouseInnerTest();
07          mit.show();
08
09      }
10      public MyMouseTest(){
11          f=new Frame("监听器");
12          tf=new TextField();
13      }
14      public void show()
15      {
16          f.add(tf,"North");
17          f.setVisible(true);
18          f.addMouseMotionListener(new TestMouseMotionListener());
19          f.setSize(300,240);
20          f.addWindowListener(new WindowAdapter(){
21              public void windowClosing(WindowEvent e) {
22                  f.setVisible(false);
23                  f.dispose();
24                  System.out.println("退出");
25                  System.exit(0);
26              }
27          });
28      }
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
29  }
30  public class TestMouseMotionListener extends MouseMotionAdapter{
31      public void mouseMoved(MouseEvent e,TextField tf){
32          tf.setText("鼠标坐标：X="+e.getX()+"Y="+e.getY());
33      }
34  }
```

【运行结果】运行结果如图 12-9 所示。



图 12-9　外部类示例效果

12.5　习题

【习题 12-1】编写一个程序，可以通过设置在窗口中的快捷键来快速打开窗口。

【分析】该题主要考查的是对事件的处理，可以设置一个键盘事件，通过输入某一个特定的快捷键来进行打开窗口操作。

【核心代码】

```
public JButtonDemo3() {
    JFrame f = new JFrame("JButtonDemo3");//创建一个 JFrame 对象，并设置 JFrame 的标题
    Container contentPane = f.getContentPane();
    JButton jb = new JButton("[o]打开新窗口");//初始化一个按钮
    jb.setMnemonic('o');                    //设置快捷键为 o
    jb.addActionListener(this);             //添加监听器
    contentPane.add(jb);
    f.pack();
    f.show();
    f.addWindowListener(new WindowAdapter() {//添加窗口监听器
        public void windowClosing(WindowEvent e) {
            System.exit(0);
        }
    });
    }
public void actionPerformed(ActionEvent e) {
        JFrame jf = new JFrame("新窗口");    //建立一个新的窗口，初始化标题为“新窗口”
        JLabel label = new JLabel("这是新窗口");//定义一个标签
        label.setHorizontalAlignment(JLabel.CENTER);
        jf.getContentPane().add(label);
        jf.setSize(100, 100);
        jf.show();
    }
```

【习题 12-2】编写一个程序，在窗口中显示出当前对按钮的操作。

【分析】该题主要考查的是对鼠标事件的处理。读者可以在窗口中设置一个按钮，并为该按钮添加事件监听，将对该按钮的操作输出到文本框中显示。

【核心代码】

```
public MouseTest() {
        jfarme = new JFrame("鼠标事件示例");                    //创建一个框架
        Container contentPane = jfarme.getContentPane();      //获得一个窗口容器
        contentPane.setLayout(new GridLayout(2, 1));          //设置布局方式
        button1 = new JButton("按钮");                        //创建一个按扭
    label = new JLabel("起始状态，还没有鼠标事件", JLabel.CENTER);//创建一个文本标签
        button1.addMouseListener(this);              //为按扭添加事件监听
        contentPane.add(label);
        contentPane.add(button1);
        jfarme.pack();                               //将窗口调节到适当的大小
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
        jfarme.show();                                  //显示窗口
        jfarme.addWindowListener(this);                 //添加窗口事件监听
    }
    public void mousePressed(MouseEvent e) {            //激发按下鼠标事件
        label.setText("你已经按下鼠标按钮");
    }
    public void mouseReleased(MouseEvent e) {           //激发释放鼠标按键
        label.setText("你已经抬起鼠标按钮");
    }
    public void mouseEntered(MouseEvent e) {            //鼠标进入组件时调用
        label.setText("光标已经进入按钮");
    }
    public void mouseExited(MouseEvent e) {             //鼠标离开组件时调用
        label.setText("光标已经离开按钮");
    }
    public void mouseClicked(MouseEvent e) {            //鼠标在组件上单击时调用
        label.setText("你单击按钮");
    }
    public void windowClosing(WindowEvent e) {
        System.exit(0);
    }
    public static void main(String[] args) {
        new MouseTest();
    }
```

【习题 12-3】将对窗口的操作输出到控制台中。

【分析】该题主要考查的是对窗口事件类的使用。读者可以添加一个窗口事件，并在控制台输出相应的操作进行显示。

【核心代码】

```
class MyWindowListener  implements WindowListener //窗体监听类继承自 WindowListener
{
    public void windowActivated(WindowEvent arg0) //重写 windowActivated 方法
    {
    }
    //窗体关闭后会调用该方法
    public void windowClosed(WindowEvent arg0)    //重写 windowClosed 方法
    {
        System.out.println("窗口关闭后");
    }
    //窗体关闭时会调用该方法
    public void windowClosing(WindowEvent arg0)      //重写 windowClosing 方法
    {
        System.out.println("窗口关闭");
        System.exit(0);
    }
    //当窗体不在活动时调用该方法
    public void windowDeactivated(WindowEvent arg0) //重写 windowDeactivated 方法
    {
    }
    //窗口从最小化状态变为正常状态时调用
    public void windowDeiconified(WindowEvent arg0) //重写 windowDeiconified 方法
    {
        System.out.println("窗口恢复");
    }
    //窗体最小化时候调用该方法
    public void windowIconified(WindowEvent arg0)   //重写 windowIconified 方法
    {
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



```
            System.out.println("最小化");
        }
        //窗口首次变为可见时调用
        public void windowOpened(WindowEvent arg0)  //重写 windowOpened 方法
        {
            System.out.println("窗口打开");
        }
}
```

【习题 12-4】编写一个程序，当拖动滚动轴时，在窗体显示当前滚轴的刻度。

【分析】该题主要考查的是对事件的处理。

【核心代码】

```
public class AdjustmentListenerDemo {
    public static void main(String[] args) {
        JFrame f = new JFrame("AdjustmentListenerDemo");
        Container c = f.getContentPane();
        final JLabel l = new JLabel("刻度：", JLabel.CENTER);
        final JScrollBar js = new JScrollBar(JScrollBar.VERTICAL, 10, 10, 0, 100);
        js.setUnitIncrement(1);
        js.setBlockIncrement(10);
        js.addAdjustmentListener(new AdjustmentListener(){
            public void adjustmentValueChanged(AdjustmentEvent e) {
                if ((JScrollBar)e.getSource() == js) {
                    l.setText("垂直刻度" + e.getValue());
                }
            }
        });

        c.add(l, BorderLayout.NORTH);
        c.add(js, BorderLayout.EAST);
        f.setSize(200, 200);
        f.setVisible(true);
    }
}
```

【习题 12-5】编写一个程序，给菜单按钮添加快捷键。

【分析】该题主要考查的是对键盘事件的处理。

【核心代码】

```
public class Hotkey {
    public static void main(String[] args) {
        JFrame f = new JFrame("Hotkey");
        Container c = f.getContentPane();

        JMenuBar m = new JMenuBar();
        JMenu file = new JMenu("文件");
        file.setMnemonic('F');
        JMenu edit = new JMenu("编辑");
        edit.setMnemonic('E');
        JMenu format = new JMenu("格式");
        format.setMnemonic('M');
        JMenu help = new JMenu("帮助");
        help.setMnemonic('H');

        m.add(file);
        m.add(edit);
        m.add(format);
        m.add(help);
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
        JMenuItem newFile = new JMenuItem("新建");
        newFile.setAccelerator(KeyStroke.getKeyStroke('N', KeyEvent.CTRL_DOWN_MASK));
        newFile.addActionListener(new ActionListener(){
            public void actionPerformed(ActionEvent e) {
                System.out.println("====");
                JFrame jf = new JFrame();
                jf.setSize(300, 300);
                jf.setVisible(true);
            }
        });
        JMenuItem open = new JMenuItem("打开");
        JMenuItem close = new JMenuItem("关闭");
        JMenuItem exit = new JMenuItem("退出");
        file.add(newFile);
        file.add(open);
        file.add(close);
        file.addSeparator();
        file.add(exit);

        f.setJMenuBar(m);

        f.setSize(200, 200);
        f.setVisible(true);
    }
}
```

【习题 12-6】编写一个程序，将窗口的变化通过窗口事件在控制台输出。

【分析】该题主要考查的是对窗体事件的处理。

【核心代码】

```
public class WindowEventDemo {
    public static void main(String[] args) {
        JFrame f = new JFrame("Window event");
        Container c = f.getContentPane();

        c.add(new Label("print window event"));

        f.addWindowListener(new WindowListener() {
            public void windowOpened(WindowEvent e) {
                System.out.println("打开窗口");
            }

            public void windowClosing(WindowEvent e) {
                System.out.println("正在关闭窗口");
            }

            public void windowClosed(WindowEvent e) {
                System.out.println("关闭窗口");
            }

            public void windowIconified(WindowEvent e) {
                System.out.println("最小化窗口");
            }

            public void windowDeiconified(WindowEvent e) {
                System.out.println("恢复窗口");
            }

            public void windowActivated(WindowEvent e) {
                System.out.println("激活窗口");
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



```
            }

            public void windowDeactivated(WindowEvent e) {
                System.out.println("未激活窗口");
            }
        });

        f.setSize(200, 200);
        f.setVisible(true);
    }
}
```

【习题 12-7】编写一个程序，通过按钮改变窗体的颜色。

【分析】该题主要考查的是对动作事件的处理。

【核心代码】

```
public class ChangeColor {
    public static void main(String[] args) {
        final JFrame f = new JFrame();
        final Container c = f.getContentPane();
        JMenuBar m = new JMenuBar();
        JButton red = new JButton("红色");
        JButton green = new JButton("绿色");
        JButton yellow = new JButton("黄色");

        m.add(red);
        m.add(green);
        m.add(yellow);

        red.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                c.setBackground(Color.red);
                c.repaint();
            }
        });
        green.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                c.setBackground(Color.green);
                c.repaint();
            }
        });
        yellow.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                c.setBackground(Color.yellow);
                c.repaint();
            }
        });

        f.setJMenuBar(m);
        f.setSize(200, 200);
        f.setVisible(true);
        f.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                System.exit(0);
            }
        });
    }
}
```

【习题 12-8】编写一个程序，当文本框获得焦点时，背景色变为红色。

【分析】该题主要考查的是对焦点事件的处理。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

【核心代码】

```
public class ChangeColorWhenFocus {
    public static void main(String[] args) {
        JFrame f = new JFrame("change color");
        final Container c = f.getContentPane();
        c.setLayout(new BorderLayout());
        JPanel p = new JPanel();

        JTextField t1 = new JTextField(10);
        JTextField t2 = new JTextField(10);
        JTextField t3 = new JTextField(10);
        FocusListener fl = new FocusListener(){
            public void focusGained(FocusEvent e) {
                ((JTextField)e.getSource()).setBackground(Color.cyan);
                ((JTextField)e.getSource()).repaint();
            }
            public void focusLost(FocusEvent e) {
                ((JTextField)e.getSource()).setBackground(Color.white);
                ((JTextField)e.getSource()).repaint();
            }
        };
        t1.addFocusListener(fl);
        t2.addFocusListener(fl);
        t3.addFocusListener(fl);

        p.add(t1);
        p.add(t2);
        p.add(t3);

        c.add(p);
        f.setSize(400, 300);
        f.setVisible(true);
        f.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                System.exit(0);
            }
        });
    }
}
```

【习题 12-9】编写一个程序，根据组合框的选择，改变文本框的内容。

【分析】该题主要考查的是对动作事件的处理。

【核心代码】

```
public class ChangeText {
    public static void main(String[] args) {
        JFrame f = new JFrame("Register");
        Container c = f.getContentPane();

        JPanel p = new JPanel();
        p.setLayout(new BorderLayout());
        String[] items = {"望庐山瀑布", "早发白帝城"};
        final String[] poets = {
                "日照香炉生紫烟，遥看瀑布挂前川。\\n 飞流直下三千尺，疑是银河落九天。",
                "朝辞白帝彩云间，千里江陵一日还。\\n 两岸猿声啼不尽，轻舟已过万重山。"
        };
        final JComboBox comb = new JComboBox(items);
        p.add(comb, BorderLayout.NORTH);

        final JTextArea t = new JTextArea();
        p.add(t, BorderLayout.CENTER);
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
        comb.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {
                t.setText(poets[comb.getSelectedIndex()]);
            }
        });

        c.add(p);

        f.setSize(500, 300);
        f.setVisible(true);
        f.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                System.exit(0);
            }
        });
    }
}
```

【习题 12-10】编写一个程序，如果双击文本区域，则文本区域底色变为灰色。

【分析】该题主要考查的是对鼠标事件的处理。

【核心代码】

```
public class DoubleClick {
    public static void main(String[] args) {
        JFrame f = new JFrame("DoubleClick");
        Container c = f.getContentPane();

        JPanel p = new JPanel();
        p.setLayout(new BorderLayout());


        final JTextArea t = new JTextArea();

        p.add(new JScrollPane(t), BorderLayout.CENTER);
        t.addMouseListener(new MouseAdapter() {
            public void mouseClicked(MouseEvent e) {
                int c = e.getClickCount();
                System.out.println(c);
                if (c == 2) {
                    t.setBackground(Color.gray);
                }
            }
        });

        c.add(p);

        f.setSize(500, 300);
        f.setVisible(true);
        f.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent
        e) {
                System.exit(0);
            }
        });
    }
}
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第 13 章　数 字 处 理

Java 中的一些数学运算可以使用 Math 类，在该类中提供了一些静态方法，可以使用这些方法来完成一些数学运算。本节主要介绍 Math 中的一些方法，并且使用 Math 类中的随机数来完成一个猜数字的游戏。



13.1　Math 类介绍

在 Math 类中提供了一些常用的数学运算方法，如正弦运算、余弦运算等，为程序提供了比较方便的数学运算操作，其中包含的字段有以下两个。

- public static final double E：用一个 double 值来表示自然对数的底数 e。
- public static final double PI：用一个 double 值来表示 PI 值，即圆的周长与直径之比。

Math 提供了一些函数，使用它们可以方便地进行数学运算。这些方法都是静态的，可以用类名直接调用，使用起来很方便，如表 13-1 所示。

表 13-1　Math 类方法

方 法 名	说　明
public static double sin(double d)	求角度的三角正弦值，注意参数为弧度表示的角度，而不是直接的角度
public static double cos(double d)	返回角度的三角余弦值，角度以弧度表示
public static double tan(double d)	返回角度的正切值，角度用弧度表示
public static double asin(double d)	返回角的反正弦值
public static double acos(double d)	返回角的反余弦值
public static double atan(double d)	返回角的反正切值
public static double toRadians(double d)	将用度数表示的角度转换为用弧度表示的值。该转换过程通常会有误差
public static double toDegrees(double d)	将用弧度表示的角度转换为用度数表示
public static double exp(double d)	返回 e 的 double 次幂的值
public static double log(double d)	返回参数的自然对数值，底数是 e
public static double log10(doule d)	返回参数的底数为 10 的对数值
public static double sqrt(double d)	返回参数值的平方根
public static double cbrt(double d)	返回参数值的立方根
public static double ceil(double d)	返回最小的大于参数的整数值
public static double floor(double d)	返回最大的小于参数的整数值
public static double rint(double d)	返回与参数值最接近的整数值，以 double 形式返回。如果两个整数都同样接近则返回偶数，如 Math.rint(6.5)的返回值为 6.0
public static double atan2(double x,double y)	将矩形坐标转换为极坐标。(x, y)转换成极坐标(r, theta)，返回值为 theta 部分
public static double pow(double d1,double d2)	返回 d1 的 d2 次幂值
public static int round(float f)	返回 f 只入不舍的 int 值
public static long round(double d)	返回 d 只入不舍的 long 值
public static double random()	返回一个 double 类型的伪随机数，范围为 0.0～1.0
public static int abs(int a)	返回 a 的绝对值

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

续表

方　法　名	说　　明
public static int max(int a,int b)	返回 a 和 b 的最大值
public static double ulp(double d)	返回参数的 ulp 值
public static float ulp(float f)	返回参数的 ulp 值
public static double signum(double d)	返回参数的符号值，如果参数大于 0，返回 1.0，参数为 0 的话返回 0，参数小于 0 返回-1.0
public static double sinh(double d)	返回 d 的双曲线正弦
public static double cosh(double d)	返回 d 的双曲线余弦
public static double tanh(double d)	返回 d 双曲线正切
public static double hypot(double x,double y)	返回 $\sqrt{x^2+y^2}$
public static double expm1(double x)	返回 e^x-1
public static double log1p(double d)	返回参数 d 与 1 的和的自然对数。当 d 等于 e-1 的时候，值为 1



13.2　猜数字游戏

本小节是一个猜数字的小程序，需要使用到 Math 中的 random 方法来产生一个随机数，然后让用户输入猜测的数字，并根据用户所输入的数字进行提示，直到用户猜出正确的数字。

【范例 13-1】示例代码 13-1 是一个猜数字的小程序。

示例代码 13-1

```
01  public class GameTest
02  {
03      public static void main(String[] args)
04  {
05          System.out.println("猜数字游戏!! ");
06          int Guess = (int) MathTest();              //产生随机数
07          int Num;                                   //输入数字
08          boolean right = true;                      //判断结果
09          while (right)                              //知道猜中数字循环结束
10              {
11                  // 提示用户输入一个整型数字。
12                  System.out.println("请输入一个整型数字 1-100 之间:");
13                  Scanner scan=new Scanner(System.in);
14                  Num=scan.nextInt();
15                  System.out.println("您输入的数字为: " + Num);
16                  // 判断用户输入的数是否正确
17                  if (Num > Guess) {
18                      System.out.println("您输入的数大于这个数；请继续努力! ");
19                  }
20                  else if (Num < Guess)
21                  {
22                      System.out.println("您输入的数小于这个数；请继续努力! ");
23                  }
24                  else
25                  {
26                      System.out.println("恭喜您猜对啦! ");
27                      System.out.println("这个数就是:" + Guess);
28                      right = false;
29                  }
30              }
31          }
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
32
33        /**
34         * 用于产生一个随机数
35         *
36         * @return 随机数
37         */
38        public static double MathTest() {
39            double MathTest = Math.random() * 100;
40            return MathTest;
41        }
42    }
```

【运行结果】程序运行结果如下。

```
猜数字游戏!!
请输入一个整型数字 1-100 之间:
50
您输入的数字为：50
您输入的数大于这个数!
请输入一个整型数字 1-100 之间:
40
您输入的数字为：40
您输入的数大于这个数!
请输入一个整型数字 1-100 之间:
30
您输入的数字为：30
您输入的数大于这个数!
请输入一个整型数字 1-100 之间:
20
您输入的数字为：20
您输入的数大于这个数!
请输入一个整型数字 1-100 之间:
10
您输入的数字为：10
您输入的数大于这个数!
请输入一个整型数字 1-100 之间:
5
您输入的数字为：5
您输入的数小于这个数!
请输入一个整型数字 1-100 之间:
7
您输入的数字为：7
恭喜您猜对啦!
这个数就是:7
```

【代码解析】以上程序中，首先要使用 Math 类创建一个随机数（第 06 行），然后使用输出语句提示用户输入一个数字（第 12～15 行）。使用 if 语句来判断用户所输入的数字，和产生的随机数做对比，并输出相应的结果（第 17～23 行）。如果用户输入的数字和产生的随机数相同，则输出“恭喜您猜对啦!”，并输出随机数（第 26～27 行）。

13.3 判断是否是数字

本节实现一个 Java 类，判断一个字符串是否是数字，主要通过下面三种方式进行判断。

- isNum(String s)，通过 Java 提供的 Character.isDigit()方法进行判断，当发现一个非数字的字符时，返回 false。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



- isNumByRegex(String s)，通过正则表达式进行判断一个字符串是否由数字组成。
- isNumByAscii(String s)，通过查看字符的 ASCII 码是否在 48～57 之间，判断是否为数字。

【范例 13-2】示例代码 13-2 是一段判断一个字符串是否是数字的程序。

示例代码 13-2

```
01  public class NumTool {
02     public static boolean isNum(String s) {
03        for (int i = s.length(); --i >= 0;) {
04           if (! Character.isDigit(s.charAt(i))) { // 通过 Java 函数判断是否是数字
05              return false;
06           }
07        }
08        return true;
09     }
10     public static boolean isNumByRegex(String s) {
11        Pattern p = Pattern.compile("[0-9]*");     // 编译正则表达式
12        return p.matcher(s).matches();
13     }
14     public static boolean isNumByAscii(String s) {
15        for (int i = s.length(); --i >= 0;) {
16           int c = s.charAt(i);
17           if (c < 48 || c > 57) {                 // 判断字符 c 是否在 48～57 之间
18              return false;
19           }
20        }
21        return true;
22     }
23     // 不提倡使用此方法
24     public static boolean isNumByException(String s) {
25        try {
26           Integer.parseInt(s);
27        } catch (NumberFormatException e) { // 捕捉 NumberFormatException
28           return false;
29        }
30        return true;
31     }
32     public static void main(String[] args) {
33        String s1 = "789";
34        System.out.println(s1 + " 是数字吗："+ isNum(s1));
35        String s2 = "667";
36        System.out.println(s2 + " 是数字吗："+ isNumByRegex(s2));
37        String s3 = "567";
38        System.out.println(s3 + " 是数字吗：" + isNumByAscii(s3));
39        String s4 = "abc";
40        System.out.println(s4 + " 是数字吗："+ isNum(s4));
41        String s5 = "555dd";
42        System.out.println(s5 + " 是数字吗："+ isNumByRegex(s5));
43        String s6 = "gh567";
44        System.out.println(s6 + " 是数字吗：" + isNumByAscii(s6));
45     }
46  }
```

【运行结果】程序运行结果如下。

```
789 是数字吗：true
667 是数字吗：true
567 是数字吗：true
abc 是数字吗：false
555dd 是数字吗：false
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
gh567 是数字吗：false
```

【代码解析】在上面的程序中，通过三种方法判断一个字符串是否是数字，isNum()逐个字符进行判断（第 02～09 行），isNumByRegex()利用正则表达式进行判断（第 10～13 行），isNumByAscii()利用 ASCII 码进行判断（第 14～22 行）。

另外，还有一种方式是通过使用方法 Integer.parseInt(s)，将一个字符串转换成数字，监控是否抛出 NumberFormatException 异常，来判断此字符串是否为数字。

```
// 不提倡使用此方法
public static boolean isNumByException(String s) {
   try {
      Integer.parseInt(s);
   } catch (NumberFormatException e) { // 捕捉 NumberFormatException
      return false;
   }
   return true;
}
```

在实际应用中，并不提倡使用此方法，因为抛出异常将消耗非常大的资源，对效率影响较大。虽然仅仅捕捉一次异常对性能的影响并不明显，但如果频繁出现异常，将是非常致命的。

13.4 字符串转换成数字

在主函数 main()中或是通过 Web 表单接收到的参数都是字符串，很多情况下需要转化成数字，下面的类提供了几个工具方法。

【范例 13-3】示例代码 13-3 是测试字符串转换成数字的程序。

示例代码 13-3

```
01  public class ConvertStr {
02     public static int convInt(String s) {          //将字符串转换为整形数字
03        try {
04           return Integer.parseInt(s);
05        } catch (NumberFormatException e) {
06           return 0;
07        }
08     }
09     public static Long convLong(String s) {        //将字符串转换为长整形
10        try {
11           return Long.parseLong(s);
12        } catch (NumberFormatException e) {
13           return 0l;
14        }
15     }
16     public static Double convDouble(String s) {  //将字符串转换为双精度类型
17        try {
18           return Double.parseDouble(s);
19        } catch (NumberFormatException e) {
20           return 0d;
21        }
22     }
23     public static int[] convIntArray(String[] as) { //将一个字符串数组转换为整形
24  //数组
25        if (as == null) {
26           return new int[0];
27        }
28        int[] ia = new int[as.length];
29        for (int i=0; i<as.length; i++) {
30           ia[i] = convInt(as[i]);
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
        }
        return ia;
    }
    public static void main(String[] args) {
        String s1 = null;
        System.out.println(s1 + " is converted into " + convInt(s1));
        String s2 = "789";
        System.out.println(s2 + " is converted into Integer: " + convInt(s2));
        String s3 = "9223372036854775807";
        System.out.println(s3 + " is converted into Integer: " + convInt(s3));
        System.out.println(s3 + " is converted into Long: " + convLong(s3));
        String s4 = "789.3";
        System.out.println(s4 + " is converted into Integer: " + convInt(s4));
        System.out.println(s4 + " is converted into Double: " + convDouble(s4));
        String[] as = null;
        convIntArray(as);
    }
}
```

【运行结果】程序运行结果如下。

```
null is converted into 0
789 is converted into Integer: 789
9223372036854775807 is converted into Integer: 0
9223372036854775807 is converted into Long: 9223372036854775807
789.3 is converted into Integer: 0
789.3 is converted into Double: 789.3
```

【代码解析】上面的工具类还提供了一个将字符串数组转换为整形数组的方法，读者可以按照自己的需要实现其他类型的转换。



13.5　格式化数字

数字的应用是多方面的，很多地方需要特殊格式的数字，比如显示百分比或以货币形式显示数字等。

【范例 13-4】示例代码 13-4 介绍了数字格式化的一般方法。

示例代码 13-4

```
public class ConvertStr {
    public static int convInt(String s) {                    //将字符串转换为整形数字
        try {
            return Integer.parseInt(s);
        } catch (NumberFormatException e) {
            return 0;
        }
    }
    public static Long convLong(String s) {                  //将字符串转换为长整形
        try {
            return Long.parseLong(s);
        } catch (NumberFormatException e) {
            return 01;
        }
    }
    public static Double convDouble(String s) {              //将字符串转换为双精度类型
        try {
            return Double.parseDouble(s);
        } catch (NumberFormatException e) {
            return 0d;
        }
    }
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
23      //将一个字符串数组转换为整形 24  数组
24      public static int[] convIntArray(String[] as) {
25          if (as == null) {
26              return new int[0];
27          }
28          int[] ia = new int[as.length];
29          for (int i=0; i<as.length; i++) {
30              ia[i] = convInt(as[i]);
31          }
32          return ia;
33      }
34      public static void main(String[] args) {
35          String s1 = null;
36          System.out.println(s1 + " is converted into " + convInt(s1));
37          String s2 = "789";
38          System.out.println(s2 + " is converted into Integer: " + convInt(s2));
39          String s3 = "9223372036854775807";
40          System.out.println(s3 + " is converted into Integer: " + convInt(s3));
41          System.out.println(s3 + " is converted into Long: " + convLong(s3));
42          String s4 = "789.3";
43          System.out.println(s4 + " is converted into Integer: " + convInt(s4));
44          System.out.println(s4 + " is converted into Double: " + convDouble(s4));
45          String[] as = null;
46          convIntArray(as);
47      }
48}
```

【运行结果】程序运行结果如下。

```
格式化：666888.0
-----------------
一般格式化：666,888
货币格式化：$666,888.00
百分比格式化：66,688,800%
使用模式：0000.0 格式化后：666888.0
使用模式：####000000000.00 $ 格式化后：000666888.00 $

格式化：367.5566
-----------------
使用模式：0.00E0 格式化后：3.68E2
使用模式：#0.00% 格式化后：36755.66%
```

【代码解析】在上面的程序中，forNumber()方法对数字进行基本的格式化，如分隔千分位等（第 02～04 行），forCurrency()方法将数字格式化成货币的形式（第 05～08 行），forPercent()方法将数字格式化成百分比的形式（第 09～11 行），forPattern()方法将数字格式化成模式所要求的格式（第 12～17 行）。

DecimalFormat 是 NumberFormat 的子类，提供了更加高级的格式化功能，可以定制格式。DecimalFormat 提供有如下一些模式字符。

- 0 代表阿拉伯数字。
- #同样代表阿拉伯数字，但如果不存在则显示为 0。
- .代表小数分隔符或货币小数分隔符。
- -代表减号或负号。
- ,代表数字分组分隔符。
- E 用于分隔科学记数法中的尾数和指数。在前缀或后缀中无须加引号。
- ;用于分隔正数和负数子模式。
- %代表百分号，乘以 100 并显示为百分数。
- \u2030 代表千分号，乘以 1000 并显示为千分数。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

- ¤ (\u00A4)表示货币记号，由货币符号替换。如果两个同时出现，则用国际货币符号替换。如果出现在某个模式中，则使用货币小数分隔符，而不使用小数分隔符。
- '用于在前缀或或后缀中为特殊字符加引号，例如，"'#'#" 将 123 格式化为 "#123"。要创建单引号本身，请连续使用两个单引号："# o''clock"。

13.6　习题

【习题 13-1】生成 10 个 100 以内的随机数字。
【分析】本题主要考查 Random 类。
【核心代码】

```
public class RandomNum {
   public static void main(String[] args) {
      Random r = new Random();

      for (int i = 0; i < 10; i++) {
         System.out.println(r.nextInt(100));
      }
   }
}
```

【习题 13-2】分别输出 100 的平方根和 1000 的立方根，以及 5 的平方根的整数值。
【分析】本题主要考查 Math 类的使用。
【核心代码】

```
public class Root {
   public static void main(String[] args) {
      System.out.println("100 的平方根："+ Math.sqrt(100));
      System.out.println("1000 的立方根："+ Math.sqrt(1000));
      System.out.println("5 的平方根的整数值："+ Math.round(Math.sqrt(5)));
   }
}
```

【习题 13-3】分别求 10 的 3 次幂，3 的 5 次幂，e 的 5 次幂。
【分析】本题主要考查的是 Math 类的使用。
【核心代码】

```
public class Pow {
   public static void main(String[] args) {
      System.out.println("10 的 3 次幂："+Math.pow(10, 3));
      System.out.println("3 的 5 次幂："+Math.pow(3, 5));
      System.out.println("e 的 5 次幂："+Math.pow(Math.E, 5));
      System.out.println("e 的 5 次幂："+Math.exp(5));
   }
}
```

【习题 13-4】写一个程序，用正则表达式判断输入的字符串是否是数字。
【分析】主要考查如何判断一个字符串是否是数字。
【核心代码】

```
public class JudgeNum {
   public static void main(String[] args) {
      Scanner reader = new Scanner(System.in); // 扫描输入文本
      System.out.println("请输入：");
      String n = reader.next();

      Boolean b = Pattern.compile("[\\d]*").matcher(n).matches();
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
        if (b) {
            System.out.println(n + "是数字");
        } else {
            System.out.println(n + "不是数字");
        }
    }
}
```

【习题 13-5】写一个程序，用 ASCII 码判断输入的字符串是否是数字。
【分析】主要考查如何判断一个字符串是否是数字。
【核心代码】

```
public static boolean byAscii(String s) {
    for (int i = 0; i < s.length(); i++) {
        int c = s.charAt(i);
        if (c < 48 || c > 57) {
            return false;
        }
    }
    return true;
}
```

【习题 13-6】编写一个程序，将输入的字符串转换成 Integer 类型。
【分析】本题主要考查的是数字的转化。
【核心代码】

```
public class ToInteger {
    public static void main(String[] args) {
        Scanner reader = new Scanner(System.in); // 扫描输入文本
        System.out.println("请输入：");
        String n = reader.next();
        try {
            System.out.println(n + "转换为数字：" + Integer.parseInt(n));
        } catch (NumberFormatException e) {
            System.out.println("转换为数字出现异常");
        }
    }
}
```

【习题 13-7】编写一个程序，将输入的字符串转换成 Double 类型。
【分析】本题主要考查的是数字的转化。
【核心代码】

```
public class ToDouble {
    public static void main(String[] args) {
        Scanner reader = new Scanner(System.in); // 扫描输入文本
        System.out.println("请输入：");
        String n = reader.next();
        try {
            System.out.println(n + "转换为数字：" + Double.parseDouble(n));
        } catch (NumberFormatException e) {
            System.out.println("转换为数字出现异常");
        }
    }
}
```

【习题 13-8】编写一个程序，把输入的字符串转换为 Double 类型，然后格式化为规范格式。
【分析】本题主要考查的是数字的转化和格式化。
【核心代码】

```
public class FormatNum {
    public static void main(String[] args) {
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



```
            Scanner reader = new Scanner(System.in); // 扫描输入文本
            System.out.println("请输入：");
            String n = reader.next();
            Double d;
            try {
               d = Double.parseDouble(n);
            } catch (NumberFormatException e) {
               d = 0d;
               System.out.println("转换为数字出现异常");
            }
            System.out.println(n + "被格式化为：" + NumberFormat.getNumberInstance().
format(d));
        }
    }
```

【习题 13-9】编写一个程序，把输入的字符串转换为 Double 类型，然后格式化为货币形式和百分比形式。

【分析】本题主要考查的是数字的转化和格式化。

【核心代码】

```
public class FormatNumTwo {
    public static void main(String[] args) {
        Scanner reader = new Scanner(System.in); // 扫描输入文本
        System.out.println("请输入：");
        String n = reader.next();
        Double d;
        try {
           d = Double.parseDouble(n);
        } catch (NumberFormatException e) {
           d = 0d;
           System.out.println("转换为数字出现异常");
        }
        System.out.println(n + "被格式化为：" + NumberFormat.getCurrencyInstance
(Locale.CHINA).format(d));
        System.out.println(n + "被格式化为：" + NumberFormat.getPercentInstance
(Locale.CHINA).format(d));
        }
    }
```

【习题 13-10】编写一个程序，把输入的字符串转换为 Double 类型，然后按指定的格式进行格式化。

【分析】本题主要考查的是数字的转化和格式化。

【核心代码】

```
public class FormatNumThree {
    public static void main(String[] args) {
        String p = "####000000000.00 $";
        Scanner reader = new Scanner(System.in); // 扫描输入文本
        System.out.println("请输入：");
        String n = reader.next();
        Double d;
        try {
           d = Double.parseDouble(n);
        } catch (NumberFormatException e) {
           d = 0d;
           System.out.println("转换为数字出现异常");
        }
        DecimalFormat df = new DecimalFormat();
        df.applyPattern(p);
        System.out.println(n + "被格式化为：" + df.format(d));
    }
}
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第 14 章　异 常 处 理

Java 中的异常处理就是一个医生，负责对 Java 程序中所出现的问题进行对应的诊治。通过 Java 的异常处理机制，可以使得代码的开发变得非常轻松。本章就对 Java 中的异常处理机制进行详细的介绍。



14.1　异常概述

Java 中的异常就像人经常会生病一样，在程序运行时也会出现各种各样的错误。人生病可能是因为饮食或者是环境的问题，而程序出错可能就是由于硬件或者资源耗尽，以及用户输入导致的。异常就是在程序运行过程中所出现的不正常的现象，例如，在做除法运算时，0 是不能作为除数使用的，计算机就会通过异常来将这个信息传出，以此进行说明。Java 中的异常可以通过关键字进行控制，主要有 try、catch、throw、throws 和 finally 五个。

- try：使用在程序中有可能会出现不正常的地方，此时可以使用 try 将可能出现异常的代码段包围起来，如果 try 语句块中出现异常，那么这个异常就会被抛出。
- catch：用来捕获异常，并在这个语句块中对该异常进行处理，如输出说明语句等。
- finally：不管发生还是不发生异常，都需要执行。
- throw：用来动手引发一个异常。
- throws：声明一个方法会抛出异常。

处理异常的一般形式如下。

```
public class AbnormalTest {
    public static void main(String[] args) {
        try
        {
            //要检查是否有异常的代码
        }
        catch(Exception1 e1)
        {
            System.out.println("该处发生 Exception1 异常");
        }
        catch(Exception2 e2)
        {
            System.out.println("该处发生异常 Exception2");
        }
        finally
        {
            //必须要执行的语句
        }
    }

}
```

以上的异常形式中，首先是将可能要抛出异常的代码块使用 try 关键字包围起来，然后使用 catch 关键字对异常进行捕获，catch 后面紧跟捕获的异常类别，然后对捕获到的异常进行相应的逻辑处理，如果代码段可能会抛出多个异常，就需要使用多个 catch 进行捕获。finally 作为最后的处理逻辑，无论捕获到异常还是没有捕获到异常，都将被执行。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

14.1.1　异常发生的原因

在程序正常运行过程中，若出现了异常，作为一个稳健的程序来说，首先需要将程序切换到另一个安全的状态，并且需要允许用户可以进行其他的操作；其次是提示用户程序出错的原因，然后保存数据并终止程序。

如果需要在程序中处理异常情况，就需要确切地知道异常是由何种原因造成的，一般情况下，导致异常出现的原因有以下几种。

- 用户输入错误：用户在需要填入数据的时候，将不符合要求的数据进行了添加。例如，需要输入一个整型的数据，而用户却输入了一个字符串类型的数据，此时就应当提示用户输入的格式错误，并返回输入界面重新输入。
- 代码本身错误：进行数据运算时，数据的长度引起的错误，或者是数组越界、算法错误、空引用等问题。
- 环境因素：磁盘空间或者内存不足所引起的错误。

异常的出现多种多样，在具体处理异常的时候，应该要判断异常，针对不同的异常类型采取不同的处理措施。

14.1.2　异常处理机制

人们生病后会去找医生，当医生对病情进行诊断后，就需要对症下药，开出一个治病的药方子，通过吃药，就会将病情治愈。程序中的异常被捕获到之后，也需要传送给“医生”进行诊治，然后根据异常做出一定的处理。异常的处理过程是，由捕获异常的方法进行异常的捕获，然后用 try-catch 语句来执行流程。

14.2　异常的结构

Java 中的异常体系非常庞大，所有的异常最终都是继承自 Throwable 接口。可以将异常分为未检查异常和已检查异常。

14.2.1　异常的层次结构

Java 中的异常主要有三种，也就是 Throwable 中的三个基本子类，如图 14-1 所示，是异常的层次结构图。

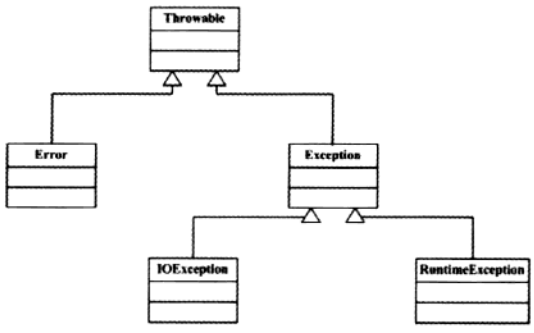


图 14-1　异常的层次结构

- Throwable：所有异常和错误的父类。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

- Error：定义 Java 运行时的内部错误以及资源耗尽错误。表示很难恢复的错误，如内存越界。一般不期望用户来进行处理。
- Exception：所有异常的父类，程序设计不合理所导致，Java 认为可以恢复的。
- RuntimeException：用来表示设计或者实现方面的问题，如数组越界、类型转换错误、去访问一个没有初始化的对象。在出现这个错误的时候，一般不需要处理，运行的时候，环境会输出一条信息语句，提示用户修正错误。
- 其他异常：表示在运行的时候，因环境因素影响可能发生并可以被处理的问题，例如，查找不到文件或者是一个不完整的路径等。因为用户的错误，很可能就会导致这类问题的发生。

在 Java 中，将任何的 Error 子类及 RuntimeException 的子类都称为未检查异常，其他的就称为已检查异常。主要的未检查异常有以下几种。

- 类型转换异常。
- 数组访问越界异常。
- 空指针访问异常。

已检查异常主要有以下几种。

- 在文件的尾部之后读取异常。
- 打开一个不存在的链接异常。
- 反射机制中查找一个不存在的类时抛出的异常。

在异常发生的时候，Java 就会将这个异常包装成一个异常类的对象，并且将其引用作为参数传递给相应的 catch 语句，这样的话，catch 语句就会对这个异常对象进行操作。

14.2.2 Java 中的运行时异常类

在 Java 中所有运行时的异常都继承自 RuntimeException 类，下面就依次介绍一下几种不同高的运行时异常。

【范例 14-1】示例代码 14-1 是一个类型转换异常的示例。

示例代码 14-1

```
01  public class ClassCastExceptionTest {
02      public static void main(String[] args) {
03      Book book=new NovelBook();
04      Object o=book;
05      String s=(String)o;
06      System.out.println(s);
07      }
08
09  }
10  class Book
11  {
14      private String title;
13      private String content;
14      private String author;
15  }
16  class NovelBook extends Book
17  {
18      private  String type="友谊";
19      public String toString()
20      {
21          return "主题："+type;
22      }
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
23   }
```

【运行结果】这个程序运行时，会弹出以下异常。

```
    Exception in thread "main" java.lang.ClassCastException: cn.com.ch14.NovelBook
cannot be cast to java.lang.String
        at cn.com.ch14.ClassCastExceptionTest.main(ClassCastExceptionTest.java:7)
```

异常 ClassCastException 就是类型转换异常的信息，因为 NovelBook 类可以被转换成 Book 类或者是 Object 类，但是不能转换成字符串类型。

【范例 14-2】示例代码 14-2 是一个访问越界的示例。

示例代码 14-2

```
01  public class ArrayIndexOutofboundsExceptionTest
02  {
03  public static void main(String[] args) {
04       int [] i=new int[]{1,2,3,4,5};
05       for(int j=0;j<6;j++)
06       {
07            System.out.println(i[j]);
08       }
09  }
10  }
```

【运行结果】程序运行结果如下。

```
1
2
3
4
5
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 5
    at cn.com.ch14.ArrayIndexOutofboundsExceptionTest.main(ArrayIndexOutofbounds
-ExceptionTest.java:9)
```

【代码解析】以上程序中所出现的 ArrayIndexOutOfBoundsException 异常就是数组访问越界异常，可以看到运行结果是在打印第 5 个数字后抛出的。

最后一种是空指针异常 NullPointerException，这种异常的出现通常是因为引用了一个没有被初始化的对象。

【范例 14-3】示例代码 14-3 是一个空指针异常的示例。

示例代码 14-3

```
01  public class NullPointerExceptionTest {
02       public static void main(String[] args) {
03            ClassTest ct=null;
04            ct.getName();
05       }
06  }
07  class ClassTest
08  {
09       private String name;            //姓名
10
11       public String getName() {
14            return name;
13       }
14
15       public void setName(String name) {
16            this.name = name;
17       }
18
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
19  }
```

【运行结果】程序运行结果如下。

```
Exception in thread "main" java.lang.NullPointerException
    at cn.com.ch14.NullPointerExceptionTest.main(NullPointerExceptionTest.java:7)
```

抛出空指针异常是因为 ClassTest 对象没有被初始化，系统还没有为其分配内存和硬件资源，所以在调用 getName()方法的时候，会由于没有可供利用的资源而抛出空指针异常。

14.3 异常的处理

在出现异常的时候，可以直接捕获处理，也可以先不对它进行处理，而是把它抛给上面的调用者。本节主要介绍 try-catch 捕获异常、throws 声明异常、throw 抛出异常、自定义异常等内容。

14.3.1 获取异常信息

捕获的异常一般都是由外界因素产生的，并且这个异常是可以恢复的，例如，一个正在网络中进行上传的文件，在上传的过程中出现文件终止传输的情况，而这个传输并不是由程序引起的。

进行异常处理的时候，可能会将抛出异常的代码放置到 try 块中，然后创建一个 catch 语句块，catch 语句块中有一个异常类型和一个代表这个类型的对象引用。如果与被抛出的异常匹配之后，catch 块中的语句就会被执行。

【范例 14-4】示例代码 14-4 是获取异常信息的示例。

示例代码 14-4

```
01  public class TryCatchTest
02  {
03      public static void main(String[] args)
04      {
05          int a,b;
06          try
07          {
08          a=0;
09          b=5/a;
10          System.out.println("检验程序...");
11          }
14          catch(ArithmeticException e)
13          {
14              System.out.println("发生 ArithmeticException 异常");
15          }
16          System.out.println("程序结束");
17      }
18  }
```

【运行结果】程序运行结果如图 14-2 所示。

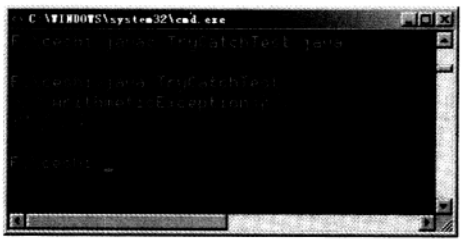


图 14-2 TryCatchTest.java 运行效果

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



从以上的运行效果中可以看出，try 语句块中的输出语句并没有被执行，之所以这样，是因为程序发生异常时，就直接从 try 执行到 catch 语句块，不再继续向下执行。执行 catch 的时候，由于发生的异常和 ArithmeticException 相吻合，从而执行了 catch 中的语句。

如果需要对异常进行处理，也可以通过 e 参数来直接告诉程序员。例如，将前面的示例代码改为下面的代码。

【范例 14-5】示例代码 14-5 是一个使用 e 参数的示例。

示例代码 14-5

```
01  public class TryCatchTest2
02  {
03      public static void main(String[] args)
04      {
05          int a,b;
06          try
07          {
08          a=0;
09          b=5/a;
10          System.out.println("检验程序...");
11          }
12          catch(ArithmeticException e)
13          {
14              System.out.println("发生的异常是："+e);
15          }
16          System.out.println("程序结束");
17      }
18  }
```

【运行结果】程序的运行效果如图 14-3 所示。



图 14-3　TryCatchTest2.java 运行效果

如果在一个 try 块中存在多个异常，就需要用到多个 catch 语句块了，可以将有可能出现的异常信息全部添加进去，如此一来，在被检查的代码块中出现异常后，就会去 catch 块中查找相匹配的异常，如果有匹配的异常，就会去执行该 catch 块中的语句。下面是一个多 catch 语句的程序。

【范例 14-6】示例代码 14-6 是一个多个 catch 语句的示例。

示例代码 14-6

```
01  public class MoreCatchTest {
02      public static void main(String[] args) {
03          int [] a=new int[3];
04          try
05          {
06          a[3]=10;
07          System.out.println("检查程序..");
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
08            }
09            catch(ArrayIndexOutOfBoundsException e)
10            {
11                System.out.println("发生 ArrayIndexOutOfBoundsException 异常");
12            }
13            catch(ArithmeticException e1)
14            {
15                System.out.println("发生 ArithmeticException 异常");
16            }
17            catch(Exception e2)
18            {
19                System.out.println("发生 Exception 异常");
20            }
21            System.out.println("程序结束");
22        }
23
24    }
```

【运行结果】程序运行结果如图 14-4 所示。



图 14-4　MoreCatchTest.java 运行效果

14.3.2　异常声明

异常声明就是指一个方法不处理它所产生的异常，而是沿调用层次向上传递，谁调用的这个方法，这个异常就由谁来进行处理。

【范例 14-7】示例代码 14-7 是一个异常声明的示例。

示例代码 14-7

```
01  public class ThrowsTest
02      {
03      public void methodName(int a)throws ArrayIndexOutOfBoundsException,
04                                          ArithmeticException{
05          System.out.println(a);
06          if(a==0)
07          {
08              System.out.println("没有异常");
09              return;
10          }
11          else if(a==1)
12          {
13              int []x=new int[3];
14              x[3]=5;
15          }
16          else if(a==2)
17          {
18              int i=0;
19              int j=50/i;
20          }
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
21        }
22        public static void main(String[] args)
23        {
24            ThrowsTest tt=new ThrowsTest();
25            try
26            {
27                tt.methodName(0);
28            }
29            catch(Exception e)
30            {
31                System.out.println("异常是: "+e);
32            }
33            try
34            {
35                tt.methodName(1);
36            }
37            catch(ArrayIndexOutOfBoundsException e)
38            {
39                System.out.println("异常是: "+e);
40            }
41            try
42            {
43                tt.methodName(2);
44            }
45            catch(ArithmeticException e)
46            {
47                System.out.println("异常是: "+e);
48            }
49        }
50    }
```

【运行结果】程序运行效果如图 14-5 所示。



图 14-5　ThrowsTest.java 运行效果

【代码解析】在以上程序中，首先在方法中声明了程序中有可能发生的所有的异常信息（第 03 行），然后在主方法中定义 try-catch 语句来进行异常信息捕获（第 25～第 48 行）。

14.3.3　抛出异常

Throw 语句用在具体方法中抛出异常，它的语法格式如下。

```
throw exception;
```

在以上的语句中，throw 是关键字，用于抛出异常。参数 exception 则是表示要抛出的异常对象，该对象应该是 Throwable 类的子类，而且只能够是一个。下面是一个在方法体中抛出异常应用的示例。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

【范例 14-8】示例代码 14-8 是一个抛出异常的示例。

示例代码 14-8

```
01  public class ThrowTest
02  {
03          public void throwException ()throws Exception
04          {
05              String s=null;
06              if(s==null)
07              {
08                  throw new Exception("s 的值为 null");//s 值为 null 时所抛出异常
09              }
10          }
11          public static void main(String[] args) {
12              ThrowTest tt=new ThrowTest();
13              try
14              {
15                  tt.throwException();  //调用 throwException 方法
16              }
17              catch(Exception e)
18              {
19                  e.printStackTrace();
20              }
21          }
22      }
```

【运行结果】程序运行结果如下。

```
java.lang.Exception: s 的值为 null
    at cn.com.ch17.ThrowTest.throwException(ThrowTest.java:10)
    at cn.com.ch17.ThrowTest.main(ThrowTest.java:17)
```

try、catch、finally 语句与 if、for 一样，也可以进行嵌套使用，也就是说一个 try-catch-finally 可以嵌套在另一个 try 语句块的 try、catch 或 finally 部分。不过，嵌套的位置不同，系统处理的流程也是不同的。

【范例 14-9】示例代码 14-9 是流程处理的示例。

示例代码 14-9

```
01  public class TryCatchFinallyTest
02  {
03          public static void main(String[] args)
04          {
05          double a=Math.random();
06          try
07          {
08              if(a>0.5)
09              {
10                  System.out.println(a);
11              }else{
12                  throw new Exception();
13              }
13          }catch(Exception el)
14          {
15              System.out.println("这是在外层所捕获的异常："+el);
16              try
17              {
18                  a=10/0;
19              }catch(ArithmeticException em)
20              {
21                  System.out.println("这是内层所捕获的异常："+em);
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
22                  }finally
23                  {
24                      System.out.println("这是内层的 finally 块");
25                  }
26              }finally
27              {
28                  System.out.println("这是外层的 finally 块");
29              }
30          }
31      }
```

【运行结果】程序运行结果如下。

```
这是在外层所捕获的异常：java.lang.Exception
这是内层所捕获的异常：java.lang.ArithmeticException: / by zero
这是内层的 finally 块
这是外层的 finally 块
```

【代码解析】以上程序中，内层的语句嵌套在外层的 catch 语句中（第 13～第 25 行），因为处理异常的时候可能会引发别的异常，所以这种写法是比较常见的。以上的运行结果不是每次都一样，这是由于开始设置的是一个随机数，所以每次产生的值是不同的。该程序运行时可能会出现如下结果。

```
0.6803348405019334
这是外层的 finally 块
```

通常在捕获到异常之后，程序员往往不知道如何去处理异常，异常堆栈信息是指发生异常的方法调用层叠的信息，如果程序运行到异常处时，可以将堆栈信息打印出来，程序员就可以根据异常的信息来判断发生异常的原因。

【范例 14-10】示例代码 14-10 是一个向上级调用传递异常的示例。

示例代码 14-10

```
01  public class ExceptionStackTest {
02      public static void main(String[] args) throws Exception {
03          ExceptionStackTest est=new ExceptionStackTest();
04          est.wrapThrowException();
05          est.ThrowException();
06      }
07      public void topThrowException ()throws Exception
08      {
09          topThrowException();
10      }
11      public void wrapThrowException ()throws Exception
12      {
13          ThrowException();
14      }
15      public void ThrowException ()throws Exception
16      {
17          throw new Exception();
18      }
19  }
```

【运行结果】程序运行结果如下。

```
Exception in thread "main" java.lang.Exception
    at cn.com.ch14.ExceptionStackTest.ThrowException(ExceptionStackTest.java:19)
    at    cn.com.ch14.ExceptionStackTest.wrapThrowException(ExceptionStackTest.
java:15)
    at cn.com.ch14.ExceptionStackTest.main(ExceptionStackTest.java:6)
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

14.4 用户自定义异常类

前面介绍了 Java 本身自带的一些异常类，但是在某些特定的情况下，还需要用户自己去创建一个异常类，创建异常类只需要继承 Throwable 类或者是它的子类 Exception 就可以了。本节主要介绍一下如何使用自定义的异常类。

14.4.1 创建自己的异常类

异常类的创建非常简单，只需要继承 Exception，并实现一些方法就可以了。创建异常类的一般形式如下。

```
class 类名 extends Exception
{
  //类体
}
```

查看 Java 的 API 可以发现，Exception 并没有定义任何方法，它从 Throwable 继承了一些方法，所以创建自定义的异常类时可以继承 Throwable 中的方法。Throwable 主要的方法如下。

- public Throwable fillInStackTrace()：该方法的作用就是返回一个包含有完全堆栈追踪的 Throwable 对象，这个对象可以被再次抛出。
- public Throwable getCause()：返回此 Throwable 对象的原因，如果原因不存在或未知，则返回 null。
- public String getMessage()：返回此 Throwable 对象的详细消息字符串。
- public StackTraceElement[] getStackTrace()：返回一个包含异常堆栈追踪的数组，每个元素表示一个堆栈帧。数组的第零个元素（假定数据的长度为非零）表示堆栈顶部，它是序列中最后的方法调用，通常，这是创建和抛出该 Throwable 对象的地方。数组的最后元素（假定数据的长度为非零）表示堆栈底部，它是序列中第一个方法调用。
- public void printStackTrace()：将异常堆栈追踪输出到标准错误流。
- public void printStackTrace(PrintStream s)：将此 Throwable 对象及其追踪输出到指定的输出流。
- public void printStackTrace(PrintWriter s)：将此 Throwable 对象及其追踪输出到指定的文本输出流。
- public String toString()：返回一个包含异常描述的 String 对象。

这些方法会被继承，也可以在创建的异常类中重写这些方法。

【范例 14-11】示例代码 14-11 是一个自定义异常类的示例。

示例代码 14-11

```
01  class MyException extends Exception{    //定义自己的异常类
02      //构造函数
03      MyException()
04      {
05      }
06      //待参数的构造函数
07      MyException(String msg)
08      {
09          super(msg);        //调用父类的方法
10      }
11  }
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



```
12  public class TestMyException{
13       public static void main(String args[]){
14    MyException mec=new MyException("这是我自定义的异常类"); //创建自己的异常类对象
15    System.out.println(mec.getMessage());                  //调用 getMessage 方法
16    System.out.println(mec.toString());
17       }
18  }
```

【运行结果】程序运行输出如下结果。

```
这是我自定义的异常类
MyException: 这是我自定义的异常类
```

14.4.2　使用自己的异常类

上一小节介绍了如何创建自己的异常类，本小节将讲解如何使用自己创建的异常类。看下面的这个例子。

```
01  class IllegalAgeException extends Exception{     //自定义异常类继承 Exception 类
02    //创建异常的构造方法
03       IllegalAgeException()
04    {
05    }
06    //创建异常的有参数的构造方法
07       IllegalAgeException(String msg)
08   {
09       super(msg);                                  //调用父类中的构造方法
10    }
11  }
12  //在测试类中使用自己定义的异常类
13  public class AgeTest{
14   public static void main(String args[])
15   {
16        //调用的方法可能会抛出异常
17        try
18        {
19            String name="张三";                     //创建姓名变量
20            String name2="李四";
21            String level=null;
22         level=AgeLevel(20);
23         System.out.println(name+"是: "+level);
24         level=AgeLevel(140);
25         System.out.println(name2+"是: "+level);
26        }
27       catch(IllegalAgeException e)
28       {
29           e.printStackTrace();
30           }
31       }
32       //定义的方法可能会产生异常
33       static String AgeLevel(int age) throws IllegalAgeException
34       {
35           if(age>=10&&age<18)
36             return "小孩";
37            else if(age>=18&&age<30)
38             return "小伙子";
39            else if(age>=30&&age<40)
40             return "中年人";
41            else if(age>=40&&age<100)
42             return "老年人";
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
43              else
44                //抛出异常
45                throw new IllegalAgeException("非法的年龄");
46          }
47      }
```

【运行结果】程序输出结果如下。

```
张三是：小伙子
cn.com.ch14.IllegalAgeException: 非法的年龄
    at cn.com.ch14.AgeTest.AgeLevel(AgeTest.java:46)
    at cn.com.ch14.AgeTest.main(AgeTest.java:25)
```

【代码解析】在以上程序中，首先定义了一个异常类 IllegalAgeException（第 01～11 行）。AgeTest 类则是对年龄进行分类的程序，将各个年龄段的成绩分为不同的名称（第 13～45 行）。其中需要考虑的是，年龄是有范围的，一般在 0～100 岁之间，所以在此规定年龄超过 100 则认为是超出范围，则会抛出相应的异常，也就是 IllegalAgeException，然后使用该异常进行处理。

14.5 习题

【习题 14-1】编写一个程序来说明多个 catch 语句顺序的重要性。

【分析】该题主要考查的是对 catch 语句的使用，首先要在程序中使用多个异常，并依次使用 catch 语句获得。

【核心代码】

```
int array[] = new int[5];
        int n = 0;
        for (int i = 0; i < array.length; i++) {
            array[i] = i;
        }
        for (int i = 0; i < 10; i++) {      // 该循环中可能会出现数组索引越界异常
            try {
                try {
                    n = array[i] / (i - 1);    // 这里可能会出现除 0 异常
                } catch (ArithmeticException e) {
                    System.out.println("0 不能做被除数");
                }
                System.out.println(n);
            } catch (IndexOutOfBoundsException e) {
                System.out.println("数组索引越界");
            }
        }
```

【习题 14-2】创建一个自定义异常类，并使用该异常。

【分析】给题主要考查对自定义异常的使用。首先要定义一个异常类，并在测试类中抛出该异常。

【核心代码】

```
// 创建一个异常类
class userException extends Exception {
    public userException(String message) { //根据指定的异常信息自定义 userException
        super(message);
    }
}
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



【习题 14-3】使用 throw 抛出一个异常 NumberFormatException。

【分析】该题主要考查对异常的抛出。

【核心代码】

```
public static void main(String argv[]) {
    try {
    ThrowException(); // 由于ThrowException声明会抛出异常，所以编译器会强制程序员在此捕获异常
    } catch (Exception e) {
        System.out.println("这是由 ThrowException 方法抛出\n" + "在 main 方法中捕获的
异常: " + e);
    }
}
public static void ThrowException() {        // 这里还是可以声明要抛出异常
    double a = Math.random();
    if (a < 0.5)
        System.out.println(a);
    else
        throw new NumberFormatException();
}
```

【习题 14-4】编写一个程序，在其中自定义两个异常类并使用。要求一个是 CustomException 类，它继承了 Exception 类，另一个是 FormatThrowable 类，它继承了 Throwable 类。

【分析】该题主要考查对自定义异常的使用。

【核心代码】

```
public static void main(String[] args) {
        try {
            customException();
            formatThrowable();
        } catch (CustomException e1) {      // 捕获 CustomException 异常
            System.out.println("Exception: " + e1.getMessage());
            e1.printStackTrace();           // 输出异常信息
        } catch (FormatThrowable e2) {      // 捕获 FormatThrowable 异常
            System.out.println("Exception: " + e2.getMessage());
            e2.printStackTrace();           // 输出异常信息
        }
    }
    public static void customException() throws CustomException {
        Throwable cause = new Throwable("继承 Exception 类的 CustomException()方法
发生异常!");
        throw new CustomException("发生错误!", cause);
    }
    public static void formatThrowable() throws FormatThrowable {
        throw new FormatThrowable("继承 Throwable 类的 throwException()方法发生异
常!");
    }
}
class CustomException extends Exception {        // 该类继承父类 Exception
    public CustomException() {                   // 默认构造方法
        super();                                 // 继承父类的的默认方法
    }
    public CustomException(String message) {     // 含有指定详细消息的构造方法
        super(message);
    }
    public CustomException(Throwable cause) {    // 含有指定的原因的构造方法
        super(cause);
    }
    public CustomException(String message, Throwable cause) {// 含有指定详细消息和
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
                                                                    //指定的原因的构造方法
        super(message, cause);
    }
}
class FormatThrowable extends Throwable {          // 该类继承 Throwable 类
    public FormatThrowable() {                     // 默认构造方法
        super();                                   // 继承父类的的默认方法
    }
    public FormatThrowable(String message) {       // 含有指定详细消息的构造方法
        super(message);
    }
    public FormatThrowable(Throwable cause) {      // 含有指定的原因的构造方法
        super(cause);
    }
    public FormatThrowable(String message, Throwable cause) {// 含有指定详细消息和
                                                                    //指定的原因的构造方法
        super(message, cause);
    }
}
```

【习题 14-5】编写一个运行时异常，并在程序中使用。

【分析】该题主要考查自定义运行时异常的使用。

【核心代码】

```
public final class BadInputException extends RuntimeException {
   public BadInputException() {
      super();
   }
   public BadInputException(String msg) {
      super(msg);
   }
}

public class TestException {
   public static void main(String[] args) {
      int x = 0;
      divide(x);
   }

   public static int divide(int den) {
      int num = 1000;
      if (den == 0) {
         throw new BadInputException();
      }
      return num / den;
   }
}
```

【习题 14-6】将一个已检查异常转化为运行时异常抛出。

【分析】该题主要考查对异常的转换。

【核心代码】

```
public class ConvertException {
   public static void main(String[] args) {
      try {
         throwCheckedException();
      } catch (ClassNotFoundException e) {
         throw new BadInputException();
      }
   }
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
    public static void throwCheckedException() throws ClassNotFoundException {
        throw new ClassNotFoundException();
    }
}
```

【习题 14-7】自定义一个已检查异常，并打印异常信息。

【分析】该题主要考查对自定义异常的使用。

【核心代码】

```
public class InvalidException extends Exception {
    public InvalidException() {
        super();
    }

    public InvalidException(String msg) {
        super(msg);
    }

    public InvalidException(Throwable cause) {
        super(cause);
    }

    public InvalidException(String msg, Throwable cause) {
        super(msg, cause);
    }
}

public class TraceException {
    public static void main(String[] args) {
        try {
            throwException();
        } catch (InvalidException e) {
            e.getMessage();
            e.getCause();
        }

    }

    public static void throwException() throws InvalidException {

    }
}
```

【习题 14-8】编写一个两个数相加的程序，如果两数的和小于其中任何一个数，就发生一个自定义异常，并输出错误信息。

【分析】该题主要考查对异常的使用。

【核心代码】

```
public class SumTwoNum {
    public static void main(String[] args) {
        int a = -9;
        int b = 4;
        int c = a + b;
        if (c < a || c < b) {
            throw new BadInputException("两数之和小于其中任何一个数");
        }
    }
}
```

【习题 14-9】编写一个程序，在一个方法的签名上声明抛出异常，主方法在调用这个方法时需要捕捉这个异常。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

【分析】该题主要考查对异常的使用。

【核心代码】

```
public class DeclareException {
    public static void main(String[] args) {
        try {
            declare();
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        }
    }

    public static void declare() throws ClassNotFoundException { //声明抛出异常
        throw new ClassNotFoundException();
    }
}
```

【习题 14-10】编写一个程序，主函数接收一个英文字符串参数，将此字符串转化成大写，如果接收到的参数为 null，则抛出 IllegalArgumentException 异常。

【分析】该题主要考查对异常的使用。

【核心代码】

```
public class ArgumentExceptionTest {
    public static void main(String[] args) {
        Scanner reader = new Scanner(System.in); //扫描输入文本
        System.out.println("请输入小写英文字符串：");
        String n = reader.next();
        if (null == n) {
            throw new IllegalArgumentException();
        }
        System.out.println(n.toUpperCase());
    }
}
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第 15 章　文件的输入与输出

文件的输入输出是程序语言设计中的一个比较重要的部分。输入输出主要面对的是对文件内容进行操作，如从键盘输入数据、从一个文件中获取到信息或者将信息写入到文件中等。本章就主要介绍一下各种输入输出流。



15.1　流的概念

在面向对象语言中，数据的输入和输出都是通过数据流来实现的。数据流是一组有顺序、有起点和终点的字符集合。就好比是两个不同的池子，一个池子中存满了水，而另一个池子中则没有任何的东西，在这两个水池中安放一个管子，水就可以从一个池子流向另一个池子了。在从一个池子向另一个池子输送水的过程中，水扮演的角色就是数据流。

数据流可以分为输入流（InputStream）和输出流（OutputStream）。输入流只可以读取数据而不能写出数据，输出流只能写入数据而不能读取数据。这就像是在水池上安装两个不同的管子，分别由阀门控制，一个管子只能够流入，另一个管子只能够流出。

在 Java 中，数据流类存在于 java.io 包中。数据流按照数据的单位可划分为字节流和字符流，其中，字节流是用来处理字节的输入和输出的，各种数据都可以使用字节来表示；字符流是用来处理字符的输入和输出的，它只可以处理文本化的数据，任何涉及文本的数据类型都应该使用字符流来进行处理。



15.2　输入输出流

可以读取字节的类就被称为输入流，而且所有的输入流都继承自抽象类 InputStream。一个能够将字节流传送出去的类被称为输出流，所有的输出流都继承自抽象类 OutputStream。输入流和输出流可以分为字节流类和字符流类。

字节流类包括 InputStream 和 OutputStream 两个抽象类。其中包含了字节输入和输出所有的方法。真正用来做数据输入和输出处理的是这两个抽象类的各个具体的子类，这些子类可以根据不同的输入和输出进行处理。输入字节流类如下。

- FileInputStream：文件字节输入流。
- ByteArrayInputStream：字节数组输入流。
- PipedInputStream：管道输入流。
- BufferedInputStream：缓冲输入流。
- FilterInputStream：过滤字节输入流。
- ObjectInputStream：对象输入流。
- DataInputStream：包含有读取 Java 标准数据类型方法的输入流。

输出字节流类如下。

- FileOutputStream：文件字节输出流。
- ByteArrayOutputStream：字节数组输出流。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

- PipedOutputStream：管道输出流。
- BufferedOutputStream：缓冲输出流。
- FilterOutputStream：过滤字节输出流。
- ObjectOutputStream：对象输出流。
- DataOutputStream：包含有读取 Java 标准数据类型方法的输出流。
- PrintStream：包含有 print()和 println()的输出流。

以上的这些类都是 InputStream 和 OutputStream 的子类，它们可以根据字节输入流和字节输出流来创建数据流，以完成所需要数据的输入和输出操作。

字符流类包括 Reader 和 Writer 两个抽象类。这些抽象类处理的是 Unicode 字符流。用来做数据处理的是 Reader 和 Writer 的子类。这些子类会根据不同的输入和输出进行处理。

输入字符流类。

- FileReader：文件字符输入流。
- CharArrayReader：从字符数组读取的输入流。
- StringReader：读取字符串的输入流。
- PipedReader：管道字符输入流。
- BufferedReader：缓冲输入流。
- FilterReader：过滤字符输入流。
- InputStreamReader：将字节输入流转化为字符输入流。

输出字符流类。

- FileWriter：文件字符输出流。
- CharArrayWriter：从字符数组输出流。
- StringWriter：字符串的输出流。
- PipedWriter：管道字符输出流。
- BufferedWriter：缓冲输出流。
- FilterWriter：过滤字符输出流。
- InputStreamWriter：将字节输出流转化为字符输出流。

以上字符输入流类和字符输出流类都是抽象类 Reader 和 Writer 类的子类。可以通过这些字符输入流类和字符输出流类来创建数据流以完成数据的输入和输出。



15.3 字节流

能够一次读取一个字节流的类是输入流，也就是 InputStream。流在 Java 中就是数据源与目的地之间的数据流动的抽象。

15.3.1 字节输入、输出流

InputStream 是字节输入流，也是所有字节输入流的超类，在它的派生类中必须要重新定义字节输入流中所声明的抽象方法。其中有一些方法，如 read()方法，是从输入流中来读取一个字节的内容，并且返回类型是整型。字节输入流中的主要方法如表 15-1 所示。

表 15-1 字节输入流方法

方 法	说 明
int read()	作用是读取一个字节，并以整数形式返回

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

续表

方　法	说　明
int read(byte b[])	作用是读取多个字节并放在字节数组 b 中，返回为数组长度
int read(byte b[], int off, int len)	作用是从字节数据的指定下标开始读取 len 个字节，返回值为实际读取字节数
int available()	作用是返回流中尚未读取的字节数
long skip(long n)	作用是读指针跳过 n 个字节，并返回实际跳的长度
close()	该方法的作用是操作完毕，关闭流
void mark(int readlimit)	该方法的作用是在输入流中标记当前的位置
void reset()	该方法作用是重新定位输入流到最后一次调用 mark 方法时的位置
boolean markSupported()	该方法的作用是判断是否支持指针记录功能

字节输出流 OutputStream 是所有字节输出流类的超类，它是一个抽象类。它的派生类必须重新定义 OutputStream 中定义的抽象方法。OutputStream 包括的方法如表 15-2 所示。

表 15-2　字节输出流方法

方　法	说　明
void write(int b)	该方法的作用是将整数写入流中
void write(byte b[])	该方法的作用是将字节数据写入流中
void write(byte b[], int off, int len)	该方法的作用是将字节数据从下标 off 开始的长度为 len 的数据写入流中
flush()	该方法的作用是刷新输出流，并输出缓冲区的字节
close()	操作完毕，关闭流

注意：通常情况下，由于 InputStream 或 OutputStream 的方法都比较低级，所以很少直接使用，而是通过实现它们的子类，来实现更高级的操作，使输入和输出更便捷。

15.3.2　字节文件输入、输出流

文件字节输入输出流是 FileInputStream 和 FileOutStream，这是 InputStream 和 OutputStream 的子类。使用这两个类就可以从一个指定的文件中读取或者向某一个文件中写入数据。

如果创建 FileInputStream 或者是 FileOutputStream 对象实例，必须要指定文件的路径和文件名，对象被创建之后，流就会被打开。程序的结尾如果不再使用流操作，必须要关闭流，以释放资源。

创建了一个 FileInputStream 对象时，指定文件必须是存在而且是可读的。而在 FileOutputStream 中，如果指定的文件已经存在，那么这个文件将会被删除，也就是会被覆盖掉，如果文件不存在，就会创建一个新的文件。

FileInputStream 类提供了方便的字节读取方法，使用 read()方法每次可以读取一个字节，并且会返回 int 类型的值。或者是使用 read()方法将读取到的数据存放到一个 byte 数组中，读取的字节个数要取决于数组的长度。在读取过程中，该 byte 数组仅作为一个缓冲区，功能相当于数据中转站。

【范例 15-1】示例代码 15-1 是一个实现输出读取文件内容的示例。

示例代码 15-1

```
01  public class InputStreamTest {
02        public static void main(String[] args) {
03              testStream();
04
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
05          }
06          /**
07           * 字节流测试
08           */
09          public static void testStream() {
10             InputStream fis = null;          //声明输入流对象
11             OutputStream fos = null;         //声明输出流对象
12             try {
13                fis = new FileInputStream("C:\\x.txt");      //设置目标文件
14                fos = new FileOutputStream("C:\\xcopy.txt");//设置输出目标文件
15                long num = 0;                 //读取字节计数
16                int bt = 0;                   //每次读入字节内容
17                       //当读入文件末尾时，读入数据的值为-1
18                   //每次读入一个字节，存放到变量 bt 中，直到读完整个文件
19                while ((bt = fis.read()) != -1) {
20                  //以字母的形式逐个输出文件的每个字节
21                   System.out.print((char) bt);
22                    fos.write(bt);   //将字节写入输出流中，实现文件的 copy 功能
23                   num++;
24                }
25           System.out.println("读取的字节数为" + num+"\n 文件复制成功"); //打印输出
26                fis.close();                      //关闭流
27                fos.close();                      //关闭流
28             } catch (FileNotFoundException e) {
29                System.out.println("找不到指定的文件！");      //打印输出异常
30                e.printStackTrace();
31             } catch (IOException e) {
32                System.out.println("文件读取时发生 IO 异常！"); //打印输出异常
33                e.printStackTrace();
34             }
35          }
36    }
```

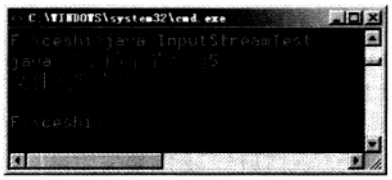


图 15-1　InputStreamTest.java 运行结果

【运行结果】程序运行结果如图 15-1 所示。

【代码解析】在程序中，首先创建输入流和输出流（第 10 行和第 11 行）。接着读取出目标文件中的内容，将读取出的内容复制到 xcopy.txt 文件中，并返回所读取内容的字节数，将该字节数输出。然后打印输出目标文件内容的字节数（第 13～23 行）。最后将使用完毕的输入流和输出流关闭，释放资源（第 26～27 行）。

注意：FileOutputStream 默认是以创建新文件的方式启动流，如果创建的文件存在，则原文件将被覆盖。如在实例化时指定 FileOutputStream 为附加模式（构造方法第二个 append 参数为 true），如果创建的文件存在，则直接打开原文件并启动流，写入数据附加到文件末端，如果不存在则创建新的文件并启动流。

15.3.3　字节缓冲输入、输出流

前面所使用的字节输入流和字节输出流是以字节为单位进行读写操作的，这样一来，就使得文件流的效率变得非常低。这时需要使用缓冲输入流（BufferedInputStream）和缓冲输出流（BufferedOutputStream），当输入数据时，将数据存放到一个缓冲区中，当需要对数据进行读写时，就直接去访问缓冲区。当输出数据时，数据不会直接向输出流输出，而是先将数据写入缓

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



冲区，当缓冲区中的数据满时，才会将缓冲区中的数据写入输出流中，这样一来就大大提高了输入和输出操作的效率。

BufferedInputStream 是 InputStream 的子类，是由 InputStream 类增加了缓冲区的功能。需要通过一个 InputStream 类型的实例来构建一个 BufferedInputStream 实例。BufferedOutputStream 是 OutputStream 的子类，和 BufferedInputStream 一样，是由 OutputStream 类增加了缓冲区的功能，创建实例的方法和 BufferedInputStream 相同。

BufferedInputStream 和 BufferedOutputStream 提供了一个数据成员 Buf（Buf 是一个数组，BufferedInputStream 提供的 Buf 默认大小为 2048 字节，而 BufferedOutputStream 提供的 Buf 默认大小为 512 字节），在程序调用 read()方法时，先去读取 Buf 中的数据，而不直接读取数据，当 Buf 中的数据不足时，才会去读取源文件数据，这正如我们喝水并不是直接从水桶里喝，而是把水倒进杯子，然后端着杯子喝水，杯子就相当于一个缓冲区，也就是我们提到的 Buf。

BufferedInputStream 和 BufferedOutputStream 常用的构造方法如下。

- BufferedInputStream(obj)
- BufferedOutputStream(obj)

在这两个构造方法中，参数 obj 是已经创建好的输入输出流对象。

【范例 15-2】示例代码 15-2 是一个使用缓冲输入流和缓冲输出流进行文件内容的复制的示例。

示例代码 15-2

```
01  public class BufferedTest
02  {
03      public static void bufferedtest()
04          {
05          FileInputStream fis = null;          //声明文件输入流
06          FileOutputStream fos = null;         //声明文件输出流
07          BufferedInputStream bis = null;      //声明缓冲输入流
08          BufferedOutputStream bos = null;     //声明缓冲输出流
09          String  name1="java.txt";            //设置读取目标文件
10          String name2="copyjava.txt";         //设置写入目标文件
11          int i;
12          try {
13              fis = new FileInputStream(name1);
14              bis = new BufferedInputStream(fis);
15
16              fos = new FileOutputStream(name2);
17              bos = new BufferedOutputStream(fos);
18
19              i=bis.read();                    //读取方法
20              while(i!=-1){
21                  bos.write(i);                //写入方法
22                  i=bis.read();
23              }
24
25          } catch (FileNotFoundException e) {
26              // TODO Auto-generated catch block
27              e.printStackTrace();
28          } catch (IOException e) {
29              // TODO Auto-generated catch block
30              e.printStackTrace();
31          }
32
33      }
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
34        public static void main(String[] args) //主方法
35        {
36            bufferedtest();                    //调用操作方法
37        }
38    }
```

【代码解析】在以上程序中，首先声明文件输入流、输出流和缓冲输入流、输出流（第 05～08 行）。然后确定读取的目标文件和写入的目标文件（第 09 和 10 行）。接下来将目标文件的路径传入，并执行 read 方法和 write 方法（第 13～22 行）。

注意：为了保证缓存区的数据可以全部被写入目的地，强烈建议在关闭流之前执行 flush()方法，将缓存区数据全部写入目的流中。

15.3.4 字节数据输入、输出流

在流操作中，还有专门针对于 Java 基本数据类型的写入和读取的操作，分别是字节数据输入流 java.io.DataInputStream 和字节数据输出流 java.io.DataOutputStream，使用字节数据流进行基本的写入和读取操作时，可以不必担心不同平台之间数据大小差异的问题，因为 Java 的数据类型占用空间的大小都是有规定的。

下面是一个使用字节数据输入流和输出流的示例，使用 DataInputStream 与 DataOutputStream 来读取 People 中的数据成员。

People.java 作为实体类，其中声明了 name 和 age 属性。

【范例 15-3】示例代码 15-3 是一个字节数据输入流、输出流的示例。

示例代码 15-3

```
01  public class People {
02      private String name;
03      private int age;
04      public int getAge() {
05          return age;
06      }
07      public void setAge(int age) {
08          this.age = age;
09      }
10      public String getName() {
11          return name;
12      }
13      public void setName(String name) {
14          this.name = name;
15      }
16      public People(String name, int age) {
17          super();
18          this.name = name;
19          this.age = age;
20      }
21  }
```

【代码解析】在以上代码中，首先声明了 name 和 age 属性（第 02 和 03 行）。然后创建属性各自的 set、get 方法（第 04～15 行）。最后创建 People 的有参构造方法（第 16～19 行）。

DataIOStreamTest 类是字节数据输入输出流的操作类，其代码如下。

```
01  public class DataIOStreamTest {
02      public static void main(String[] args) {
03          String filename = "f:\\FileTest\\DataioStreamTest.txt";
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
04  //创建，并初始化文件名字符串
05          //创建，并初始化 People 类型数组
06          People[] people = {
07              new People("赵子龙",23),
08              new People("关云长",24),
09              new People("曹操",25),
10              new People("刘备",22)};
11          try {
12           DataOutputStream dataOutputStream =new DataOutputStream(
13        new FileOutputStream(filename));          //创建 DataOutputStream 对象
14           for(People Employee : people)          //使用 for 变量
15           {
16              // 写入字符串
17              dataOutputStream.writeUTF(Employee.getName());
18              // 写入数据
19              dataOutputStream.writeInt(Employee.getAge());
20           }
21           // 读出所有数据至目的地
22           dataOutputStream.flush();
23           // 关闭流
24           dataOutputStream.close();
25           DataInputStream dataInputStream = new DataInputStream(
26                    new FileInputStream(filename));
27           // 读出数据并还原为对象
28           for(int i = 0; i < people.length; i++) {
29              // 读出 UTF 字符串
30              String name = dataInputStream.readUTF();
31              // 读出 int 数据
32              int score = dataInputStream.readInt();
33              people[people.length-1 -i] = new People(name, score);
34           }
35           // 关闭流
36           dataInputStream.close();
37           // 输出还原后的数据
38           for(People Employee : people) {
39             System.out.printf("%s\t%d%n",
40                    Employee.getName(), Employee.getAge());     //格式化输出
41           }
42        }
43        catch(IOException e) {        //捕获异常
44           e.printStackTrace();       //异常信息输出
45        }
46     }
47  }
```

【运行结果】程序运行结果如图 15-2 所示。

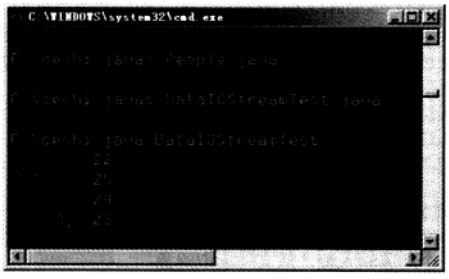


图 15-2　DataIOStreamTest.java 运行结果

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

【代码解析】在以上程序中，首先定义一个字符串类型，用于接收文件路径，接下来创建 People 并初始化对象（第 06～10 行），并将数据写入（第 14～19 行），使用 readUTF()或 readInt() 等方法读取出数据（第 28～34 行）。

15.3.5 字节对象输入、输出流

字节对象输入、输出流也就是对象序列化，对象序列化就是指将继承了 Serializable 接口的类的实例转换成一个字节序列，并且可以将其存储在永久性存储介质中。在需要的时候可以将其从存储介质中取出并还原成具体的示例来进行使用。Java 提供了字节对象输入流 java.io.ObjectInputStream 和字节对象输出流 java.io.ObjectOutputStream 来实现这样的操作。

类 ObjectOutputStream 负责将对象序列化，而类 ObjectInputStream 则负责将序列化的对象还原。在序列化时，首先是要选择一个要存放文件的路径，因此还需要建立一个 FileOutputStream 实例，并且要作为参数传入 ObjectOutputStream。



注意：对象序列化是指对象能够将自己的状态信息数据保存下来的特性。对象序列化的目的是使得程序中的一个对象经过存储或者网络传输之后，另外一个程序可以将对象还原。

对于对象序列化，需要注意以下几点。

- 只有实现 Serializable 接口的对象才可以被序列化。没有实现此接口的类将无法使其任何状态序列化或反序列化。
- private void writeObject (Object obj)：该方法主要作用是输出特定类的对象的状态。状态是通过使用 writeObject 方法，将各个字段写入 ObjectOutputStream 来保存的。
- private void readObject(ObjectInputStream in)：该方法的主要作用是从 ObjectInputStream 流中读取并还原类字段。
- 序列化时，对象按照 writeObject 方法的调用顺序存储在文件中，先被序列化的对象的数据在文件的前面，后被序列化的对象的数据在文件的后面。因此，在反序列化时，先读到的对象肯定是先被序列化的对象。

【范例 15-4】示例代码 15-4 是一个实现序列化的示例。

示例代码 15-4

```
01  public class ObjectioTest {
02      // 将对象序例化
03      public static void serialize(String fileName) throws Exception {
04          // 创建一个对象输出流，将对象输出到文件
05          ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream(
06                  fileName));
07          out.writeObject("今天的日期和时间如下："); // 序列化一个字符串对象到文件
08          out.writeObject(Calendar.getInstance());// 序列化当前日期对象到文件
09          // 序列化一个 MyClass 对象
10          TestClass tc = new TestClass(1, 2, 3, 4);
11          out.writeObject(tc);
12          out.close();
13      }
14      // 将序列化的对象反序列
15      public static void deserialize(String fileName) throws Exception {
16          // 创建一个对象输入流，从文件读取对象
17          ObjectInputStream in = new ObjectInputStream(new FileInputStream(
18                  fileName));
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
19            // 读取字符串对象
20            String today = (String) (in.readObject());
21            System.out.println(today);
22            // 读日历 Calendar 对象
23            Calendar date = (Calendar) (in.readObject());
24            System.out.println(date.get(Calendar.YEAR) + "年"
25                    + (date.get(Calendar.MONTH) + 1) + "月"
26                    + date.get(Calendar.DAY_OF_MONTH) + "日: "
27                    + date.getTime().toLocaleString());
28            // 从输入流中读取 TestClass 对象。
29            TestClass tc = (TestClass) (in.readObject());
30            System.out.println(tc.toString());
31            in.close();
32        }
33        public static void main(String[] args) {
34            System.out.println("反序列化的结果如下: ");
35            String fileName = "D:/temp/TestClass.txt";
36            try {
37                ObjectioTest.serialize(fileName);
38            } catch (Exception e) {
39                System.out.println("对象序列化失败" + e.getMessage());
40            }
41            try {
42                ObjectioTest.deserialize(fileName);
43            } catch (Exception e) {
44                System.out.println("对象反序列化失败" + e.getMessage());
45            }
46        }
47    }
48    // 一个类，用于被序列化和反序列化。
49    //一定要实现 Serializable 才能够被序列化和反序列化。
50    class TestClass implements Serializable {
51        private int i, n; // 一般的实例变量会被序列化和反序列化
52        private transient int x; // 用 transient 声明的变量不会被序列化和反序列化
53        private static int y; // 类变量也不会被序列化和反序列化
54        public TestClass(int m, int n, int x, int y) {// 利用构造方法赋值
55            this.i = m;
56            this.n = n;
57            this.x = x;
58            TestClass.y = y;
59        }
60        public String toString() {
61            return i + " : " + n + "  :" + x + "  :" + y;
62        }
63    }
```

【运行结果】程序运行结果如图 15-3 所示。

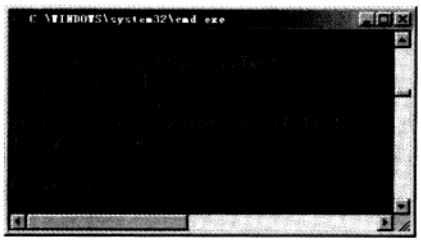


图 15-3　ObjectioTest.java 运行结果

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

【代码解析】以上程序中，首先创建对象输出流，然后传入路径参数，并序列化字符串“今天的日期是”到文件中，同时序列化一个时间到文件中（第 03～12 行）。然后再定义一个反序列化类，用于反序列化取出字符串和当前的日期（第 15～31 行）。创建主方法，并在主方法中设置路径并调用相关方法，返回相关序列化的信息（第 33～44 行）。

注意：ObjectOutputStream 在写入对象数据时，会加入一个特别的流头（Stream Header），目的是为了 ObjectInputStream 在读出的时候检查这个流头。如果一个文件被附加对象，那么这个文件会有多个流头，这会导致读取文件检查时发生错误，抛出 java.io.StreamCorrupedException 异常。为了解决这个问题，可重新定义 ObjectOutputStream 的 writeStreamHeader()方法，在以附加方式写文件时，不再写流头。

```
ObjectOutputStream objOutputStream =
   new ObjectOutputStream(new FileOutputStream(file, true)) {
     protected void writeStreamHeader() throws IOException {}
   };
```

序列化对象的完整读写功能并不仅仅应用于文件操作，也可应用在网络数据传输等领域。

15.4 字符流

字符输入输出流是 Reader 和 Writer，Reader 类是字符输入流的基类，其中包括有 FileReader、BufferedReader 等子类。Writer 是字符输出流的基类。使用 Reader 和 Writer 对流数据的操作是以一个字符的长度为单位进行处理的，也可以进行字符的编码处理。

15.4.1 字符读、写流

字符流支持 Unicode 标准字符集，如果进行文本读写，通常在使用 Reader 和 Writer 的子类时要重新定义相关的方法。而 Reader 和 Writer 只提供了用于字符流处理的接口，而不能生成实例，只可以通过他们的子类对象来处理字符流。

Reader 是 Java 语言中字符输入流所有类的父类，该类是抽象类，不能被实例化。如下是 Reader 中定义的很多与读取字符串相关的方法。

- public int read()：读取一个字符，并返回该字符。
- public int read(char cbuf[])：读取一系列字符数组，返回字符数组长度。
- public abstract int read(char cbuf[],int off,int len)：读取从下标 off 开始，长度为 len 的字符保存到字符数组中，返回实际读取字符数。
- public boolean markSupported()：判断流是否支持标记。
- public void mark(int readAheadLimit)：标记流的当前位置。
- public void reset()：将流重置标记处。

Writer 是 Java 语言中字符输出流所有类的父类，该类是抽象类，不能被实例化。如下是 Writer 中定义的与输出字符相关的方法。

- public void write(int c)：将整型数据的低 14 位写入输出流。
- public void write(char cbuf[])：将字符数组 cbuf[]写入输出流。
- public abstract void write(char cbuf[],int off,int len)：将字符数组从索引 off 开始的 len 个字符写入输出流。
- public void write(String str)：将字符串的字符写入输出流。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



- public void write(String str,int off,int len)：将字符串的索引 off 开始的 len 个字符写入输出流。
- flush()：刷新输出，并输出在缓冲区中的字符。

15.4.2　字符输入、输出流

可以使用 Reader 和 Writer 的子类 InputStreamReader 和 OutputStreamWriter 来进行字符的处理，也就是以字符为基本单位进行读写操作，并且可以实现字符流和字节流之间的转化，字符流操作比字节流操作效率要高。

使用 InputStream 和 OutputStream 对象为构造方法的参数创建 InputStreamReader 和 OutputStreamWriter 变量可以直接读取文本，而不需要自己再判断字符编码。

【范例 15-5】示例代码 15-5 是一个实现输出读取文件内容并复制的示例。

示例代码 15-5

```
01  public class ReaderTest {
02      public static void main(String[] args) throws IOException {
03          reader();
04      }
05      public static void reader() throws IOException {//新建一个方法，名为 reader
06          /**
07           *   字符流测试
08           */
09           // 定义一个指向 C:\\x.txt 的字节流
10          FileInputStream fileInputStream = new FileInputStream("C:\\x.txt");
11          //字节流转换成 InputStreamReader
12          InputStreamReader inputStreamReader = new InputStreamReader(fileInput
            -Stream);
13         //        InputStreamReader 转换成带缓存的 bufferedReader
14          BufferedReader bufferedReader = new BufferedReader(inputStream
            -Reader);
15       // InputStreamReader 转换成带缓存的 bufferedReader
16          String ss = new String();
17          String s;              //将读取出来的信息复制给字符串 s
18
19          while ((s = bufferedReader.readLine()) != null) {
20              ss += s;
21          }
22          //定义一个指向输出的字节流
23          FileOutputStream fileOutputStream = new FileOutputStream("D:\\x.
            txt");
24          // 字节流转换成 outputStreamWriter
25          OutputStreamWriter outputStreamWriter = new OutputStreamWriter(
26                  fileOutputStream);
27          //outputStreamWriter 转换成带缓存的 bufferedReader
28          BufferedWriter bufferedWriter = new BufferedWriter(outputStream
            -Writer);
29           bufferedWriter.write(ss);
30           bufferedWriter.close();
31          outputStreamWriter.close();
32          fileOutputStream.close();
33      }
34  }
```

【运行结果】以上程序运行结果如图 15-4 所示。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



图 15-4 Reader.java.java 运行结果

【代码解析】在以上程序中，首先定义一个 FileInputStream 实例对象，并传入文件的路径（第 10 行）。然后将字节流转换成 InputStreamReader 并读取文件中的内容（第 12～14 行）。接着定义 FileOutputStream 实例对象并传入目的文件地址（第 23 行），并将读取出的文件内容写入到目的文件中（第 25～29 行）。

使用 InputStreamReader 还可以指定读取文件的编码格式，例如，指定读取文件的编码为 gb2312。

```
InputStreamReader inputStreamReader = new InputStreamReader(fileInputStream,
"gb2312");
```

15.4.3 文件读、写字符流

文件读写字符流是 FileReader 和 FileWriter，它们分别继承自 InputStreamReader 与 OutputStreamWriter，而且使用文件输入输出流可以直接指定文件名或者使用 File 来打开指定的文件。如果需要指定编码的话，就需要使用 InputStreamReader 和 OutputStreamWriter。

FileReader 中有两种形式的常用构造方法如下。

- FileReader(File file)：创建一个给定 File 文件的 FilerReader。
- FileReader(String name)：创建一个给定文件名 name 的 FilerReader。

FileReader(File file)方法内有一个 File 类型参数，此方法的作用是根据指定的文件路径创建一个 FileReader，其中的 file 参数就是指定的文件路径。

FileReader(String name)方法内有一个字符串类型的参数，此方法的作用是根据一个指定的文件名创建一个 FileReader，其中的 name 参数就是指定的文件全称。

FileWriter 中有四种形式的常用构造方法如下。

- FileWriter(File file)：创建一个指定文件路径的 FileWriter。
- FileWriter(File file,Boolean b)：创建一个指定文件路径的 FileWriter，并根据 Booblean 值区分文件的打开方式。
- FileWriter(String name) ：创建一个指定文件名的 FileWriter。
- FileWriter(String name,Boolean b)：创建一个指定文件名的 FileWriter，并根据 Booblean 值区分文件的打开方式。

FileWriter(File file)方法内有一个 File 类型参数，此方法的作用是根据指定文件路径创建一个 FileWriter，其中的 file 参数就是指定的文件路径。

FileWriter(File file，Boolean b)方法内有一个 File 类型参数，此方法的作用是根据指定文件路径创建一个 FileWriter，其中的 file 参数就是指定的文件路径，并根据其 b 的值确定如何打开文件，如果 b 的值为 true，则在文件的原有内容后追加新内容，如果 b 的值为 false，则在文件

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



的开头部分写入新内容。

FileWriter(String name)方法内有一个字符串类型参数，此方法的作用是根据指定文件名创建一个 FileWriter，其中方法内的 name 参数就是指定的文件名。

FileWriter(String name，Boolean b)方法内有一个字符串类型参数，此方法的作用是根据指定文件名创建一个 FileWriter，其中方法内的 name 参数就是指定的文件名，并根据其 b 的值确定如何打开文件，如果 b 的值为 true，则在文件的原有内容后追加新内容，如果 b 的值为 false，则在文件的开头部分写入新内容。

【范例 15-6】示例代码 15-6 是一个在文件中追加新内容的示例。

示例代码 15-6

```
01  public class FileReaderTest
02  {
03   public static void filereadertest()
04       {
05      String name="javatest.txt";          // New 一个字符串类型的 name，并赋初值为
06  //“java.txt”
07       try {
08      FileReader fr=new FileReader(name);//创建一个指定文件名的 FileReader，并传值
                                           //name;
09          char[] c=new char[1];           //new 一个 char 类型长度为 1 的数组
10          while( fr.read(c)!=-1){         //循环判断是否读取到文件的底部
11               System.out.print(new String(c));    //将读取到的内容打印到显示器上
12          }
13
14          fr.close();                                //关闭流
15      } catch (FileNotFoundException e) {
16          // TODO Auto-generated catch block
17          e.printStackTrace();
18      } catch (IOException e) {
19          // TODO Auto-generated catch block
20          e.printStackTrace();
21      }
22   }
23    public static void main(String[] args)         //主方法
24       {
25      filereadertest();                           //调用操作方法
26      }
27  }
```

【运行结果】程序运行结果如图 15-5 所示。

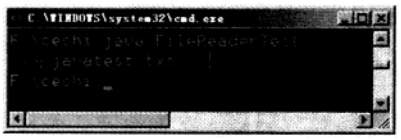


图 15-5　读取 javatest.txt 文件中的内容

【代码解析】这个程序首先创建了一个字符串类型的 name，并赋值为文件的文件名（第 05～06 行）。创建一个指定文件名的 FileReader（第 08 行）。将已经赋值的 name 传入，再创建一个字符类型的数组，并且数组的长度定义为 1（第 09 行）。用循环判断是否读取到文件的底部，如果读取到文件的底部则跳出循环，输出显示读取到的内容（第 10～11 行）。最后关闭流（第 14 行）。



注意：如果只输入文件名的话，这个文件的默认地址为项目的根目录，如果使用 FileReader(File file)的话就可以传入源文件的相对路径或绝对路径了。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

15.4.4 字符缓冲读、写流

字符缓冲读写流是 BufferedReader 和 BufferedWriter，它们的默认缓冲区为 8192 个字符。和字节缓冲读写流类似，读取文件时，需要首先将文件读取到缓冲区，然后使用 read()方法从缓冲区读取，如果缓冲区的数据不足，就会再次从文件中读取字符到缓冲区。同理，BufferedWriter 写入数据时，不会直接将数据写入到目的地，而是写入到缓冲区中。

BufferedReader 类中有如下两个构造方法，并支持 mark()和 reset()方法。

- BufferedReader(Reader in)：以 Reader 对象 in 为参数创建 BufferedReader。
- BufferedReader(Reader in,int sz)：以 Reader 对象 in 为参数创建 BufferedReader，并且可以通过参数 sz 来指定缓冲区的大小。

BufferedWriter 类中有如下两个构造方法。

- BufferedWriter(Writer out)：以 Writer 对象 out 为参数创建 BufferedWriter。
- BufferedWriter(Writer out,int sz)：以 Writer 对象 out 为参数创建 BufferedWriter，并且可以通过参数 sz 来指定缓冲区的大小。

 15.5 文件类

在输入输出处理中，对于文件的读写，需要使用到 java.io 包中有关于文件处理的类，其中包括有 File、FileInputStream、FileOutputStream、RandomAccessFile 和 FileDescriptor；接口有 FilenameFilter。

15.5.1 File 类

学习如何读取和写入数据之前，首先需要了解一个文件操作的工具类 File，使用该类既可以处理数据文件也可以处理目录，并且通过该类可以列出 File 实例所代表的目录下所有的文件名。在处理一个目录之前，首先要做的就是创建一个 File 的实例用于表示该目录，语法如下。

```
File file=new File("src");
```

在不同操作系统中的路径名称的表示也是不同的，如在 Windows 中，路径名为“C:\Program Files\Java”，而在 Linux 中的路径就是“/home/ Program Files/Java”。这是因为 Windows 中使用的是 UNC 路径名，以“\\”开始的目录表示上下文环境所在目录的硬盘根目录，如果没有以“\\”作为路径的开始，则表示相对于当前工作目录的路径，并且通过“盘符（C/D/E…）:”形式表示硬盘指定。然而在 Linux 中，并没有硬盘驱动器的概念，所以它的路径指定必须以“/”来表示根目录开始的绝对路径，其他的则是相对路径。



注意：在 Java 中，无论是文件还是目录，都使用 File 实例表示。

下面介绍类 File 中提供的各种方法。

- public File(String path)：如果其中的参数是实际存在的路径，则该对象表示的是目录，如果其中的参数 path 是文件名，则表示该 File 对象是文件。
- public File(String path,String name)：其中的参数 path 表示的是路径名，而参数 name 则表示文件名。
- public File(File dir,String name)：其中的参数 dir 表示的是路径名，而参数 name 表示的

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



则是文件名。

- String getName()：该方法用于得到一个文件的名称，但其中不包括路径。
- String getPath()：该方法用于得到一个文件的路径名。
- String getAbsolutePath()：该方法用于得到一个文件的绝对路径。
- String getParent()：该方法用于得到一个文件的上一级目录名称。
- String renameTo(File newName)：该方法用于将当前的文件名更名为所指定的文件的完整路径。
- boolean exists()：该方法用于测试当前 File 对象所制定的文件是否存在，如果存在，则返回 true，否则返回 false。
- boolean canWrite()：测试当前的文件是否可写，如果可写，则返回 true，否则返回 false。
- boolean canRead()：该方法用于测试当前的文件是否可读。
- boolean isFile()：该方法用于测试当前的文件是否是文件而不是目录。如果是文件，则返回 true，如果是目录，则返回 false。
- boolean isDirectory()：该方法用于测试当前的文件是否是目录。
- long lastModified()：该方法用于得到文件最近一次修改的时间。
- long length()：该方法用于得到文件的长度，以字节为单位。
- boolean delete()：该方法用于删除当前文件。
- boolean mkdir()：该方法用于根据当前对象生成一个由该对象指定的路径。
- String list()：列出当前目录下的文件。

15.5.2　File 类应用

下面是一个演示 File 类方法使用的例子，该程序用于遍历文件夹下面的文件，并且输出文件夹的信息，由于程序中使用了过滤器 Filter，只读取 java 文件。

【范例 15-7】示例代码 15-7 是一个实现文件过滤读取的示例。

示例代码 15-7

```
01    public class FileTest {
02        public void ReadFileList(File file,Filter filter)
03        {
04            if(file.isDirectory())                    //首先判断 file 是否为目录
05            {
06          try {
07              File[] files = file.listFiles(filter); //创建目录数组
08           ArrayList<File> fileList = new ArrayList<File>();  //通过数组创建
                                                                 //数组列表
09             for(int i = 0; i < files.length; i++) {  //for 循环变量数组
10              // 先列出目录
11              if(files[i].isDirectory()) {          //判断是否为目录
12                  System.out.println("[" +files[i].getPath() + "]");
                                                      //输出路径名
13              ReadFileList(files[i],filter);        //递归调用 ReadFileList 方法
14                  }
15                  else {
16                      // 文件先存入 fileList
17                      fileList.add(files[i]);
18                  }
19               }
20                  // 列出文件
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
21            for(File f: fileList) {                    //使用增强型 for 语句遍历数组
22                    ReadFileList(f,filter);            //调用 ReadFileList 方法
23                }
24                    System.out.println();              //输出换行符
25             }
26             catch(ArrayIndexOutOfBoundsException e) {//捕获异常
27               e.printStackTrace();                     //打印异常信息
28        }
29               }else if(file.isFile())                 //当 file 为文件时
30               {
31                    FileDesc(file);                    //调用 FileDesc 方法
32               }
33
34           }
35           public void FileDesc(File file)
36           {
37             if(file.isFile()) { // 判断是否为文件
38             System.out.print(file.toString() + " \n 该文件");    //输出字符串信息
39             System.out.print(file.canRead() ? "可读 " : "不可读 ");//判断文件是否
                                                                    //可读
40             System.out.print(file.canWrite() ? "可写 " : "不可写 "); //判断文件
                                                                      //是否可写
41             System.out.println(file.length() + "字节");          //输出字节数
42             }
43           }
44          public static void main(String[] args) {
45           String filename = "F:\\FileTest\\";     //由参数获取文件名
46           File file = new File(filename);         //创建，并初始化由文件名决定的文
                                                     //件对象
47           Filter filter = new Filter("java");     //创建，并初始化文件过滤器
48       new FileTest().ReadFileList(file,filter);   //创建 FileDemo 实例，并调用
                                                     //ReadFileList 方法
49      }
50       }
51       class Filter implements FilenameFilter{
52            String extent;
53            //声明，实现 Filter 的构造方法
54            Filter(String extent){
55                 this.extent=extent;    //初始化成员变量 extent
56            }
57            //覆盖父类的 accept 方法
58            public boolean accept(File dir,String name){
59                 return name.endsWith("."+extent);  //返回文件的后缀名
60            }
61   }
```

【运行结果】程序运行结果如图 15-6 所示。

【代码解析】在以上程序中，首先通过递归方法完成文件信息的输出，也就是 ReadFileList 方法（第 09～19 行）。然后进行判断，如果该方法中的参数是一个文件，则会调用 FileDesc 方法，然后输出文件的信息（第 35～43 行）；如果该参数是一个目录，则递归调用 ReadFileList 方法。

15.5.3 RandomAccessFile 类

RandomAccessFile 类是对随机访问文件的读取和写入。该随机访问文件类似于存储在文件

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



系统中的一个庞大的字节数组，输入操作从该数组的开始读取字节，并随着对字节的读取而前移。如果随机访问文件以读取/写入模式创建，则输出操作也可用。输出操作从文件指针开始写入字节，并随着对字节的写入而前移。写入隐含数组的当前末尾之后的输出操作导致该数组扩展。该文件指针可以通过 getFilePointer 方法读取，并通过 seek 方法设置。

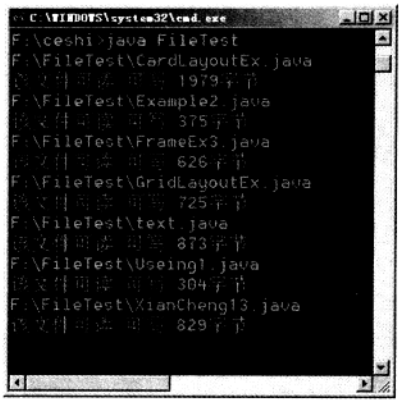


图 15-6　FileTest.java 运行结果

1．构造方法

- RandomAccessFile(String name,String mode)：name 是文件名，mode 是打开方式，如"r"表示只读、"rw"表示可读写。
- RandomAccessFile(File file,String mode)：file 是文件对象。

2．文件指针的操作

- long getFilePointer()：用于得到当前的文件指针。
- void seek(long pos)：用于移动文件指针到指定的位置。
- int skipBytes(int n)：使文件指针向前移动指定的 n 个字节。

为了保证移动存取位置的正确性，通常必须在随机文件读取之前固定每个数据的长度，例如，读取存储雇员信息的文件时，固定为每一个雇员的数据大小，Java 中并没有直接的方法取得固定长度的数据，必须自己计算获得。

15.5.4　RandomAccessFile 应用

RandomAccessFile 类在随机（相对顺序而言）读写等长记录格式的文件时有很大的优势，如在读取数据库中的某一条记录时。但是，RandomAccessFile 类仅限于操作文件，不能访问其他的 IO 设备，如网络，内存映像等。

【范例 15-8】示例代码 15-8 是一个 RandomAccessFile 示例。

示例代码 15-8

```
01  public class ReadomAccessFileTest {
02      public static void RandomToWrite(File file) {
03          Student[] students = new Student[4];
04          // 创建 Student 数组
05          // 初始化 Student 数组
06          students[0] = new Student("优秀", "90");
07          students[1] = new Student("优良", "80");
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
08          students[2] = new Student("良好", "70");
09          students[3] = new Student("及格", "60");
10          RandomAccessFile randomAccessFile = null; // 创建 RandomAccessFile 对象
11          try {
12      randomAccessFile = new RandomAccessFile(file, "rw"); // 创建一个可以支持
                                                              //读写状态的
13                                                          //RandomAccessFile 对象
14          } catch (FileNotFoundException e1) {    // 捕获异常
15          System.out.println("文件没有找到" + e1.getMessage()); // 输出异常信息
16          }
17          try {
18              for (int i = 0; i < students.length; i++) //for 遍历 students 数组
19              {
20          randomAccessFile.writeChars(students[i].getLevel());// 向文件中写入评
                                                                //分等级
21              randomAccessFile.writeChars(students[i].getScore());// 向文件中
                                                                    //写入分数
22              }
23              randomAccessFile.close();       // 关闭 randomAccessFile
24          } catch (IOException e1) {          // 捕获异常
25              e1.printStackTrace();           // 输出异常信息
26          }
27      }
28      private static String readName(RandomAccessFile randomAccessfile)
29              throws IOException {
30          char[] name = new char[5];          // 创建 char 类型数组
31          for (int i = 0; i < name.length; i++) {
32              name[i] = randomAccessfile.readChar(); // 读取字符
33          }
34          return new String(name);
35      }
36      public static Student[] RandomToRead(File file) throws Exception {
37      RandomAccessFile randomAccessFile;          // 创建 RandomAccessFile 实例
38      randomAccessFile=newRandomAccessFile(file,"r");//初始化 randomAccessFile
                                                       //对象
39  int num = (int) randomAccessFile.length() / Student.size(); // 返回 Student 类
                                                               //的占用空间大小
40      Student[] student = new Student[num];       // 创建 Student 数组 student
41      for (int i = 0; i < num; i++) {
42      randomAccessFile.seek((i) * Student.size());
43          // 使用对应的 read 方法读出数据。
44          student[i] = new Student(readName(randomAccessFile),
45              readName(randomAccessFile));
46          }
47          randomAccessFile.close(); // randomAccessFile 关闭
48          return student;             // 返回数组
49      }
50          public static void main(String[] args) throws Exception {
51  //          创建，并初始化文件名称
52          String filename = "D:/temp/a.txt";
53          File file = new File(filename);     // 创建并初始化 File 对象 file
54          RandomToWrite(file);                // 调用 RandomWriteFile 方法
55          Student[] student = RandomToRead(file); // 返回文件中保存的 student 数组
56          // 使用 for，遍历 student 数组
57          System.out.println(filename + "的内容如下：");
58          for (int i = 0; i < student.length; i++) {
59              System.out.println("Level=" + student[i].getLevel() + "|Score="
60                  + student[i].getScore());
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
61                }
62        }
63    }
64    class Student {
65        String level;      // 评分级别
66        String score;      // 分数
67        final static int SIZE = 8;                    // 创建，并初始化静态域 LEN
68        public Student(String level, String score) {
69            if (level.length() > SIZE) {
70                level = level.substring(0, 8);        // 截取字符串的子字符串
71            } else {
72                while (level.length() < SIZE)
73                    level = level + "\u0000";
74            }
75            this.level = level;    // 初始化 level
76            this.score = score;    // 初始化 score
77        }
78        // 获取类占用的空间
79        public static int size() {
80            return 20;    // 字符串长度是 8，一个字符占用 2 个字节，一个整型是 4 个字节
81        }
82        public String getLevel() {
83            return level;
84        }
85        public void setLevel(String level) {
86            this.level = level;
87        }
88        public String getScore() {
89            return score;
90        }
91        public void setScore(String score) {
92            this.score = score;
93        }
94    }
```

【运行结果】程序运行结果如图 15-7 所示。



图 15-7　ReadomAccessFileTest.java 运行结果

【代码解析】在以上程序中，要创建一个内部类作为实体类，声明 level 和 score 属性，并创建相对性的 set/get 方法（第 64～94 行）。创建并初始化 Student 数组（第 03～09 行）。使用循环遍历 Student 数组并将分数和评价写入（第 18～25 行）。

由以上可以看出，使用 RandomAccessFile 读写文件时，有以下几个必要的流程。

（1）在进行文件操作之前必须打开文件，并且需要指定文件的读写方式。

（2）写文件必须使用对应的写入方法。

（3）读文件必须使用对应的读取方法。

（4）读写完成之后关闭文件。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

15.6 习题

【习题 15-1】编写一个程序，在一个目录下创建一个文件，然后再创建一个新的空文件，并在控制台输出相应的信息。

【分析】该题主要考查文件类的使用，首先要判断所制定的目录是否存在，如果不存在则要根据路径自动创建一个，然后在该目录下创建一个新的文件夹。

【核心代码】

```
// 此方法的主要作用是判断创建文件是否成功，如果成功则返回 true 否则返回 false
public static boolean createFile(String destFileName) {
    File file = new File(destFileName);     // 根据指定的文件名创建 File 对象
    if (file.exists()) {                    // 判断该文件是否存在
        System.out.println("创建单个文件" + destFileName + "失败，目标文件已存在！");
        return false;                       // 如果存在，则不需创建则返回 fasle
    }
    if (destFileName.endsWith(File.separator)) {// 如果传入的文件名是以文件分隔符结尾
                                                //的，则说明此 File 对象是个目录而不是文件
        System.out.println("创建单个文件" + destFileName + "失败，目标文件不能为目录！");
        return false;// 因为不是文件所以不可能创建成功，则返回 false
    }
    // 判断目标文件所在的目录是否存在
    if (!file.getParentFile().exists()) {
    // 如果目标文件所在的文件夹不存在，则创建父文件夹
        System.out.println("创建" + file.getName() + "所在目录不存在，正在创建！");
        if (!file.getParentFile().mkdirs()) {// 判断父文件夹是否存在，如果存在则表
                                             //示创建成功否则不成功
            System.out.println("创建目标文件所在的目录失败！");
            return false;
        }
    }
    // 创建目标文件
    try {
        if (file.createNewFile()) {// 调用 createNewFile 方法，判断创建指定文件是否成功
            System.out.println("创建单个文件" + destFileName + "成功！");
            return true;
        } else {
            System.out.println("创建单个文件" + destFileName + "失败！");
            return false;
        }
    } catch (IOException e) {
        e.printStackTrace();
        System.out.println("创建文件" + destFileName + "失败！" + e.getMessage());
        return false;
    }
}
```

【习题 15-2】编写一个程序，从键盘上接收用户输入的数据，并将数据保存到文件中，当输入的一行字符串为“end”时，程序停止接收键盘的数据，并进行保存。

【分析】该题主要考查数据的保存。结合 BufferedReader 类和 System 类，首先获取到键盘中所输入的字符串，然后将其保存到指定的文件中，如果输入的一行字符串为“end”时，停止接受，并执行保存操作。

【核心代码】

```
// 将由键盘中输入的内容保存在指定的文件中
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



```
public static boolean saveMessageToFile(String fileName) {
    File file = new File(fileName);
    // 将数据按照文本输出到文件
    PrintWriter pw = null;
    BufferedReader in = null;
    try {
        // 为输出文件建立一个写入器
        pw = new PrintWriter(new FileWriter(file));
        System.out.println("请输入文件的内容并输入 end 结束");
        // 用 BufferedReader 包装标准输入流
        in = new BufferedReader(new InputStreamReader(System.in));
        String inputLine = null;
        while (((inputLine = in.readLine()) != null)
                && (!inputLine.equals("end"))) {
            pw.println(inputLine);
        }
        pw.flush();
        pw.close();
        return true;
    } catch (IOException e) {
        System.out.println(e.getMessage());
        return false;
    } finally {
        if (in != null) {
            try {
                in.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}
// 按照文件中格式将文件内容显示出来
public static void readFileMessage(String fileName) {
    File file = new File(fileName);
    BufferedReader reader = null;
    try {
        System.out.println("按顺序读取文件的内容如下：");
        reader = new BufferedReader(new FileReader(file));
        String string = null;
        int line = 1;
        // 按行读取内容，直到读入 null 则表示读取文件结束
        while ((string = reader.readLine()) != null) {
            System.out.println("line " + line + ": " + string);
            line++;
        }
        reader.close();
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if (reader != null) {
            try {
                reader.close();
            } catch (IOException e1) {
            }
        }
    }
}
```

【习题 15-3】编写一个程序，将程序中给定目录的多个小文件进行合并。读取出所有小文件中的内容，将其全部合并成一个大的文件。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

【分析】该题首先要确定指定目录中的文件，然后分别对各个文件进行读取，读取完毕后，将所有的内容保存到一个文件中，并删除其他的文件。

【核心代码】

```
public static void merge(String[] fileNames, String TargetFileName)
        throws Exception {
    File fin = null;
    // 构建文件输出流
    File fout = new File(TargetFileName);
    FileOutputStream out = new FileOutputStream(fout);
    for (int i = 0; i < fileNames.length; i++) {
        // 打开文件输入流
        fin = new File(fileNames[i]);
        FileInputStream in = new FileInputStream(fin);
        // 从输入流中读取数据，并写入到文件数出流中
        int c;
        while ((c = in.read()) != -1) {
            out.write(c);
        }
        in.close();
    }
    out.close();
    System.out.println("合并文件" + TargetFileName + "中的内容如下：");
}
public static void readFileMessage(String fileName) {// 将合成的文件中的内容读出
    File file = new File(fileName);
    BufferedReader reader = null;
    try {
        reader = new BufferedReader(new FileReader(file));
        String string = null;
        // 按行读取内容，直到读入 null 则表示读取文件结束
        while ((string = reader.readLine()) != null) {
            System.out.println(string);
        }
        reader.close();
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if (reader != null) {
            try {
                reader.close();
            } catch (IOException e1) {
            }
        }
    }
}
```

【习题 15-4】编写一个程序，将指定文件中的字符数统计出来，其中包括有数字字符的总数、文字字符的总数、标点符号的总数，以及最后总的字符数，并将其结果打印到控制台中进行显示。

【分析】该题首先要确定一个指定的文件，并将其内容读取出来，然后对其中的字符进行统计，并分别打印其统计结果。

【核心代码】

```
// 统计指定文件中的字符的总数，其中：fileName 指的是文件的全路径
    public static long xermote(String fileName) {
        FileReader reader = null;
        try {
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



```
                reader = new FileReader(fileName);
                // 创建分析给定字符流的标记生成器
                StreamTokenizer stn = new StreamTokenizer(
                        new BufferedReader(reader));// 根据给定的字符流创建标记生成器 stn
// 下面指定单引号、双引号和注释的转义字符在此标记生成器中是普通字符，不代表其他的特殊的含义
                stn.ordinaryChar('\'');
                stn.ordinaryChar('\"');
                stn.ordinaryChar('/');
                // 创建可以统计字符总数的各个变量
                String str;
                int num_sum = 0;  // 用于统计数字字符总数的变量
                int word_sum = 0; // 用于统计字母、汉字等字符总数的变量
                int sym_sum = 0;  // 用行统计标点符号等字符总数的变量
                int sum = 0;      // 用于计算全部种类字符总数的变量
                // 当生成器的下一个标记不是流末尾的常量
                while (stn.nextToken() != StreamTokenizer.TT_EOF) {
                    switch (stn.ttype) {// 来判断当前获取的流的标记类型
                    // TT_EOL 指示已读到行末尾的常量。
                    case StreamTokenizer.TT_EOL:
                        break;
                    // TT_NUMBER 指示已读到一个数字标记的常量
                    case StreamTokenizer.TT_NUMBER:
                    str = String.valueOf((stn.nval));// stn.nval 表示此数字标记的值
                        num_sum += str.length();
                        break;
                    // TT_WORD 指示已读到一个文字标记的常量
                    case StreamTokenizer.TT_WORD:
                        str = stn.sval;// stn.sval 表示此文字标记的值
                        word_sum += str.length();
                        break;
                    default:
                        // 如果以上 3 种类型都不是，则为英文的标点符号
                        str = String.valueOf((char) stn.ttype);
                        sym_sum += str.length();
                    }
                }
                System.out.println("数字标记的和为：" + num_sum);
                System.out.println("文字标记的和为：" + word_sum);
                System.out.println("标点符号标记的和为：" + sym_sum);
                sum = sym_sum + num_sum + word_sum;
                System.out.println("此文件中的字符总数为：" + sum);
                return sum;
            } catch (Exception e) {
                e.printStackTrace();
                return -1;
            } finally {
                if (reader != null) {
                    try {
                        reader.close();
                    } catch (IOException e1) {
                    }
                }
            }
        }
```

【习题 15-5】创建一个程序，使用 FileWriter 类和 BufferedWriter 类，将特定内容输出到文件里。

【分析】该题主要考查 FileWriter 和 BufferedWriter 的使用。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

【核心代码】

```
FileWriter fw;
        try {
            fw = new FileWriter(args[0]);                    // 创建字符输出流对象
            BufferedWriter bf = new BufferedWriter(fw); // 创建缓冲字符输出流对象
            for (int i = 0; i < 10; i++) {                   // 将字符串写入文件
                bf.write("Line" + i);
            }
            bf.close();                                      // 关闭缓冲字符输出流
        } catch (IOException e) {
            e.printStackTrace();
        }
```

【习题 15-6】创建一个程序，打印出给定目录下的子目录和文件名。

【分析】该题主要考查 File 类的使用。

【核心代码】

```
public class ExploreFolder {
   public static void main(String[] args) {
      Scanner reader = new Scanner(System.in); // 扫描输入文本
      System.out.println("请输入目录路径：");
      String name = reader.next();
      explore(new File(name));
   }
   public static void explore(File f) {
      if (!f.exists()) {
         System.out.println(f.getPath() + " 不存在。");
         return;
      }
      if (!f.isDirectory()) {
         System.out.println(f.getPath() + " 不是目录。");
         return;
      }
      File[] fs = f.listFiles();
      for (File file : fs) {
         System.out.println(file.getPath());
      }
   }
}
```

【习题 15-7】创建一个程序，使用字符流打印出给定文本文件的内容。

【分析】该题主要考查 BufferedReader 与 InputStreamReader 的使用。

【核心代码】

```
public class ReadText {
   public static void main(String[] args) {
      Scanner reader = new Scanner(System.in); // 扫描输入文本
      System.out.println("请输入文件路径：");
      String name = reader.next();
      try {
         read(new File(name));
      } catch (IOException e) {
         throw new BadInputException("读取文件内容错误"); // 异常一章创建的运行时异常
      }
   }
   public static void read(File f) throws IOException {
      if (!f.exists()) {
         System.out.println("文件不存在。");
         return;
      }
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
            if (f.isDirectory()) {
                System.out.println("这是一个目录。");
                return;
            }
            BufferedReader br = new BufferedReader(new InputStreamReader(new
FileInputStream(f), "UTF-8")); // 假设此文本文件以 UTF-8 编码保存
            String s;
            while ((s = br.readLine()) != null) {
                System.out.println(s);
            }
            br.close();
        }
    }
```

【习题 15-8】创建一个程序，将一个 Product 类序列化到文件，并反序列化。

【分析】该题主要考查 ObjectOutputStream 和 ObjectInputStream 的使用。

【核心代码】

```
public class Product implements Serializable {
    private String name;
    private String price;

    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }

    public String getPrice() {
        return price;
    }
    public void setPrice(String price) {
        this.price = price;
    }
}

public class SerializeObject {
    public static void main(String[] args) {
        Scanner reader = new Scanner(System.in); // 扫描输入文本
        System.out.println("请输入文件路径：");
        String name = reader.next();
        File f = new File(name);
        if (f.exists()) {
            System.out.println("文件已存在。");
            return;
        }
        if (f.isDirectory()) {
            System.out.println("这是一个目录。");
            return;
        }

        try {
            serialize(f);
            deserialize(f);
        } catch (IOException e) {
            throw new BadInputException("序列化文件出现错误"); // 异常一章创建的运行时异常
        } catch (ClassNotFoundException e) {
            throw new BadInputException("文件转型错误"); // 异常一章创建的运行时异常
        }
    }
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
    public static void serialize(File f) throws IOException {
        Product p = new Product();
        p.setName("特步时尚板鞋");
        p.setPrice("200");
        ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream(f));
        out.writeObject(p);
        out.flush();
        out.close();
    }

    public static void deserialize(File f) throws IOException, ClassNotFoundException {
        ObjectInputStream in = new ObjectInputStream(new FileInputStream(f));
        Product p = (Product) in.readObject();
        System.out.println("商品名称: "+ p.getName());
        System.out.println("商品价格: "+ p.getPrice());
    }
}
```

【习题 15-9】创建一个程序，将一段文字写入文本文件，并指定 UTF-8 编码。

【分析】该题主要考查 OutputStreamWriterlistener 类及 BufferedWriter 类的使用。

【核心代码】

```
public class WriteText {
    public static void main(String[] args) {
        Scanner reader = new Scanner(System.in); // 扫描输入文本
        System.out.println("请输入文件路径: ");
        String name = reader.next();
        try {
            write(new File(name));
        } catch (IOException e) {
            throw new BadInputException("写入文件内容错误");
        }
    }
    public static void write(File f) throws IOException, UnsupportedEncodingException {
        BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(new
FileOutputStream(f), "UTF-8"));
        bw.write("不到长城非好汉，屈指行程二万");
        bw.flush();
        bw.close();
    }
}
```

【习题 15-10】创建一个程序，使用 RandomAccessFile 写一段文字到一个文本文件，然后将其读出。

【分析】该题主要考查 RandomAccessFile 的使用。

【核心代码】

```
public class RandomText {
    public static void main(String[] args) {
        Scanner reader = new Scanner(System.in); // 扫描输入文本
        System.out.println("请输入文件路径: ");
        String name = reader.next();
        File f = new File(name);
        if (f.exists()) {
            System.out.println("文件已存在。");
            return;
        }
        if (f.isDirectory()) {
            System.out.println("这是一个目录。");
            return;
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



```
        }
        try {
            write(f, "清平乐 六盘山\r\n");
            write(f, "天高云淡，望断南飞雁。\r\n");
            write(f, "到长城非好汉，屈指行程二万。\r\n");
            write(f, "六盘山上高峰，红旗漫卷西风。\r\n");
            write(f, "今日长缨在手，何时缚住苍龙。");

            read(f);
        } catch (IOException e) {
            throw new BadInputException("RandomAccessFile 错误"); // 异常一章创建的运行
                                                                  //时异常
        }
    }
    public static void read(File f) throws IOException {
        RandomAccessFile ra = new RandomAccessFile(f, "r");
        long sum = ra.length();
        while (ra.getFilePointer() < sum) {
            System.out.println(ra.readUTF());
        }
        ra.close();
    }

    public static void write(File f, String s) throws IOException {
        RandomAccessFile ra = new RandomAccessFile(f, "rw");
        ra.seek(ra.length());
        ra.writeUTF(s);
        ra.close();
    }
}
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第 16 章　反　　射

反射机制可以为程序提供动态的运行能力，利用反射可以在运行时对程序进行动态的控制。Java 提供有一套专门的工具类，帮助程序员在程序中使用反射特性。本章就主要介绍一下 Java 的反射机制。



16.1　反射概述

反射是 Java 提供的比较特殊的特性，可以使用它对程序进行动态控制。开发使用反射的程序所需要的工具类在 java.lang.reflect 包中。

Class 类属于 java.lang 包，而且该类不需要使用 import 语句引入特殊的包就可以使用了。它的对象代表着一个类，携带类的相应信息，其中主要包括有构造器、方法和成员变量等。在 Java 程序的运行过程中，每一个类被加载之后，都会在内存中产生一个相对应的 Class 对象，而这些对象是由系统来自动进行维护的。



注意：该处的“Class”中的“C”是大写，如果写成小写的话，就变成类的关键字了。

在 Java 中提供了一些比较常用的方法，用来获取或者是加载相应的信息（如表 16-1 所示）。

表 16-1　Class 类常用方法

方　法	说　明
static Class<?> forName(String name, boolean initialize, ClassLoader loader)	使用给定的类加载器，返回与带有给定字符串名的类或接口相关联的 Class 对象
getAnnotation(Class<A> annotationClass)	如果存在该元素的指定类型的注释，则返回这些注释，否则返回 null
Annotation[] getAnnotations()	返回此元素上存在的所有注释
String getCanonicalName()	返回 Java Language Specification 中所定义的底层类的规范化名称
Class<?>[] getClasses()	返回一个包含某些 Class 对象的数组，这些对象表示属于此 Class 对象所表示的类的成员的所有公共类和接口
ClassLoader getClassLoader()	返回该类的类加载器
Class<?> getComponentType()	返回表示数组组件类型的 Class
Constructor<T> getConstructor(Class<?>... parameterTypes)	返回一个 Constructor 对象，它反映此 Class 对象所表示的类的指定公共构造方法
Constructor<?>[] getConstructors()	返回一个包含某些 Constructor 对象的数组，这些对象反映此 Class 对象所表示的类的所有公共构造方法
Annotation[] getDeclaredAnnotations()	返回直接存在于此元素上的所有注释
Class<?>[] getDeclaredClasses()	返回 Class 对象的一个数组，这些对象反映声明为此 Class 对象所表示的类的成员的所有类和接口

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

续表

方　法	说　明
Constructor<T> getDeclaredConstructor(Class<?>... parameterTypes)	返回一个 Constructor 对象，该对象反映此 Class 对象所表示的类或接口的指定构造方法
Constructor<?>[] getDeclaredConstructors()	返回 Constructor 对象的一个数组，这些对象反映此 Class 对象表示的类声明的所有构造方法
Field getDeclaredField(String name)	返回一个 Field 对象，该对象反映此 Class 对象所表示的类或接口的指定已声明字段
Field[] getDeclaredFields()	返回 Field 对象的一个数组，这些对象反映此 Class 对象所表示的类或接口所声明的所有字段
Method getDeclaredMethod(String name, Class<?>... parameterTypes)	返回一个 Method 对象，该对象反映此 Class 对象所表示的类或接口的指定已声明方法
Method[] getDeclaredMethods()	返回 Method 对象的一个数组，这些对象反映此 Class 对象表示的类或接口声明的所有方法，包括公共、保护、默认（包）访问和私有方法，但不包括继承的方法
Class<?> getDeclaringClass()	如果此 Class 对象所表示的类或接口是另一个类的成员，则返回的 Class 对象表示该对象的声明类
Class<?> getEnclosingClass()	返回底层类的立即封闭类
Constructor<?> getEnclosingConstructor()	如果该 Class 对象表示构造方法中的一个本地或匿名类，则返回 Constructor 对象，它表示底层类的立即封闭构造方法
Method getEnclosingMethod()	如果此 Class 对象表示某一方法中的一个本地或匿名类，则返回 Method 对象，它表示底层类的立即封闭方法
Field getField(String name)	返回一个 Field 对象，它反映此 Class 对象所表示的类或接口的指定公共成员字段
Field[] getFields()	返回一个包含某些 Field 对象的数组，这些对象反映此 Class 对象所表示的类或接口的所有可访问公共字段
Type[] getGenericInterfaces()	返回表示某些接口的 Type，这些接口由此对象所表示的类或接口直接实现
Type getGenericSuperclass()	返回表示此 Class 所表示的实体（类、接口、基本类型或 void）的直接超类的 Type
Class<?>[] getInterfaces()	确定此对象所表示的类或接口实现的接口
Method[] getMethods()	返回一个包含某些 Method 对象的数组，这些对象反映此 Class 对象所表示的类或接口（包括那些由该类或接口声明的以及从超类和超接口继承的那些类或接口）的公共 member 方法
int getModifiers()	返回此类或接口以整数编码的 Java 语言修饰符
String getName()	以 String 的形式返回此 Class 对象所表示的实体（类、接口、数组类、基本类型或 void）名称
URL getResource(String name)	查找带有给定名称的资源
InputStream getResourceAsStream(String name)	查找具有给定名称的资源
String getSimpleName()	返回源代码中给出的底层类的简称
Class<? super T> getSuperclass()	返回表示此 Class 所表示的实体（类、接口、基本类型或 void）的超类的 Class

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

续表

方　法	说　明
TypeVariable<Class<T>>[] getTypeParameters()	按声明顺序返回 TypeVariable 对象的一个数组，这些对象表示用此 GenericDeclaration 对象所表示的常规声明来声明的类型变量
boolean isArray()	判定此 Class 对象是否表示一个数组类
boolean isInstance(Object obj)	判定指定的 Object 是否与此 Class 所表示的对象赋值兼容
boolean isMemberClass()	当且仅当底层类是成员类时返回 true

16.1.1　使用 Class 类

前面介绍了 Class 类的常用方法，本小节就通过几个例子来说明 Class 的使用方法。

【范例 16-1】示例代码 16-1 说明 Class 的使用方法。

示例代码 16-1

```
01  class Father
02  {
03      //父类的公共成员变量
04      public int memberFather;
05      //父类中的公共方法
06      public void methodFather()
07      {
08          System.out.println("这是从父类中继承而来的方法 methodFather");
09      }
10  }
11  class Son extends Father
12  {
13      //子类的额公共成员变量
14      public int memberSonPublic;
15      //子类的私有成员变量
16      private int memberSonPrivate;
17      //子类的公共方法
18      public void methodSonPublic()
19      {
20          System.out.println("这是子类自己的方法 methodSonPublic");
21      }
22      public void methodSonProtected()
23      {
24          System.out.println("这是子类自己的方法 methodSonProtected");
25      }
26  }
27  public class ClassTest
28  {
29      public static void main(String[] args)
30      {
31          try
32          {
33              //加载指定的类
34              Class cs=Class.forName("cn.com.ch16.Son");
35              Son myson=(Son)cs.newInstance();
36              //调用创建对象的方法
37              System.out.println("调用创建对象的方法:");
38              myson.methodSonProtected();
39              myson.methodSonPublic();
40              myson.methodFather();
41              //加载类的详细信息
42              System.out.println("加载类的信息:");
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
43              System.out.println(cs.getName()+"自己声明了"+
44                      cs.getDeclaredFields().length+"个成员变量");
45              System.out.println(cs.getName()+"类对外公布的方法有"+
46                      cs.getMethods().length+"个");
47          }catch(Exception e){
48              e.printStackTrace();
49          }
50      }
51  }
```

【运行结果】程序运行结果如下。

```
调用创建对象的方法:
这是子类自己的方法 methodSonProtected
这是子类自己的方法 methodSonPublic
这是从父类中继承而来的方法 methodFather
加载类的信息:
cn.com.ch16.Son 自己声明了 2 个成员变量
cn.com.ch16.Son 类对外公布的方法有 12 个
```

【代码解析】在以上程序中，创建一个父类 Father 类，并创建成员变量和方法（第 01～10 行）。创建一个子类 Son，并继承 Father 类，然后在子类中创建变量和方法（第 11～26 行）。在主方法中加载指定的类（第 34～35 行）。调用所创建的对象方法（第 38～40 行）。最后打印出加载类的详细信息（第 42～46 行）。

16.1.2　成员变量

Java 反射用 Field 类表示成员变量，Field 类提供了动态访问成员变量的 API。Class 中获取成员变量的 API 如下：

- getField(String name)返回一个 Field 对象，name 参数表示此成员变量的名称，成员变量必须是 public 类型。
- getFields()返回一个 Field 对象的数组，这些成员变量必须都是 public 类型的。
- getDeclareField(String name)返回一个 Field 对象，name 参数表示此成员变量的名称，此成员变量可以是已声明的任何类型。
- getDeclaredFields()返回一个 Field 数组，这些成员变量可以是已声明的任何类型。

Field 类比较常用的方法如下。

- setAccessible(Boolean flag)方法设置 accessible 的值，true 时表示取消 Java 访问检查，false 表示启用 Java 访问检查。
- getType()方法返回一个 Class 对象，标识此字段的声明类型。
- getName()方法返回此 Field 对象标识的字段的名称。
- set(Object obj, Object value)将指定对象上的此 Field 所表示的成员变量的值设置为指定的值。
- getModifiers()以整数形式返回此 Field 对象所表示的字段的 Java 语言修饰符，请使用 Modifier 类对这些修饰符进行解码。

16.1.3　成员方法

Java 反射用 Method 类表示成员方法，Method 类提供了获取成员方法信息和调用成员方法的 API。Class 中获取成员方法的 API 如下：

- getMethod(String name, Class… parameterTypes)返回一个 Method 对象，name 参数表示

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

成员方法的名称，parameterTypes 参数是成员方法的形式参数的数组，成员方法必须是 public 类型。

- getMethods()返回一个 Method 对象数组，这些成员方法必须都是 public 类型。
- getDeclaredMethod(String name, Class… parameterTypes)返回一个 Method 对象，name 参数表示成员方法的名称，parameterTypes 参数是成员方法的形式参数的数组，此成员方法可以是已声明的任何类型。
- getDeclaredMethods()返回一个 Method 对象数组，这些成员方法可以是已声明的任何类型。

Method 类比较常用的方法如下。

getName()返回 Method 对象所表示的方法的名称。

getParameterTypes()以 Class 对象数组的形式，返回 Method 对象所表示的方法的形参类型。

getReturnType()以 Class 对象的形式，返回 Method 对象所表示的方法的返回类型。

getModifiers()以整数形式返回此 Method 对象所表示的方法的 Java 语言修饰符。

invoke(Object obj, Object… args) 用给定的参数调用给定对象的方法（此 Method 对象所代表的方法）。

16.1.4 创建实例

反射中有两种方法可以创建实例对象：无参数的构造方法和带参数的构造方法。通过无参数的构造方法创建实例比较简单，只要调用 Class 对象的 newInstance()方法即可；而调用带参数的构造方法创建实例就比较复杂，首先需要通过一个 Class[]参数获得包含此参数的构造方法，然后准备一个 Object[]作为参数调用此 Constructor 对象的 newInstance()方法。



注意：只有 Class 类和 Constructor 类拥有 newInstance()方法。Class 类中的 newInstance()方法是不带参数的，而 Constructror 类中的 newInstance()方法是带参数的。

利用反射实例化一个类的有关方法如下。

- getConstructors()获得类的 public 类型的构造方法。
- getConstructor(Class[] parameterTypes)获得类的特定构造方法，parameterTypes 参数指定构造方法的参数类型。
- newInstance()通过类的不带参数的构造方法创建这个类的一个对象。



16.2 应用反射

16.2.1 访问类变量

通过 Java 的反射机制，可以轻松地获得类变量的所有信息，并且可以修改变量的值。如果这些属性是 private 或 protected 的，则必须设置 setAccessible (true)方法，否则会抛出 IllegalAccessException 异常，如果启用 Java 安全机制，则可能会抛出 SecurityException 异常。

【范例 16-2】示例代码 16-2 演示了如何通过反射访问和更改类变量的方法。

示例代码 16-2

```
01  public class House {
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
    public String owner;
    protected int space;
    private int price;
    public String address;
    private static String name;
    public House() {
    }
    public House(String owner, int space, int price, String address, String n)
{
        this.owner = owner;
        this.space = space;
        this.price = price;
        this.address = address;
        name = n;
    }
}
public class FieldTest {
    public static void main(String[] args) throws IllegalAccessException {
        House house = new House("George", 160, 300, "Beijing", "Happy House");
        Class c = house.getClass();
        Field[] fields = c.getDeclaredFields(); // 取出所有成员变量
        System.out.println("共有成员变量: "+ fields.length);
        System.out.println("------------------");
        for (Field field : fields) {
            field.setAccessible(true);  // 设置此变量的可访问性为 true
            Class t = field.getType();  // 取得此变量的类型
            if (t == String.class) {
                System.out.println(field.getName() + " : " + field.get(house));
            }
            if (t == int.class) {
                System.out.println(field.getName()+": "+field.getInt(house));
// 取得 int 类型的值
            }
        }
        System.out.println("\n 为变量设置新值: ");
        System.out.println("----------------");
        for (Field field : fields) {
            field.setAccessible(true);
            String n = field.getName();             // 取得此变量的名称
            if (n.equalsIgnoreCase("owner")) {
                field.set(house, "George Li");      // 为此变量设置新值
            }
            if (n.equalsIgnoreCase("space")) {
                field.set(house, field.getInt(house) * 2);
            }
            if (n.equalsIgnoreCase("price")) {
                field.set(house, field.getInt(house) * 2);
            }
            if (n.equalsIgnoreCase("address")) {
                field.set(house, "Beijing China");
            }
            if (n.equalsIgnoreCase("name")) {
                field.set(house, "Brilliant House");
            }
        }
        for (Field field : fields) {
            System.out.println(field.getName() + " : " + field.get(house));
        }
    }
}
```

【运行结果】程序运行结果如下。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
共有成员变量：5
------------------
owner : George
space : 160
price : 300
address : Beijing
name : Happy House

为变量设置新值：
----------------
owner : George Li
space : 320
price : 600
address : Beijing China
name : Brilliant House
```

【代码解析】在以上程序中，利用 House 的实例获得了 Class 对象。通过 Class 对象取得的 House 的所有属性，并将变量的名字和值打印出来。然后通过设置 setAccessible (true)方法，修改 House 公共变量及私有变量的值。

16.2.2 调用类方法

同样，通过 Java 反射机制，也可以很轻松地调用方法进行运算。私有方法和被保护的方法也需要设置 setAccessible (true)方法，避免 Java 安全机制抛出异常。

【范例 16-3】示例代码 16-3 演示了通过反射调用类方法的方法。

示例代码 16-3

```
01  public class Radio {
02     private String brand() {
03        System.out.println("A famous brand: YongSheng");
04        return "YongSheng";
05     }
06     protected void turnOn() {
07        System.out.println("The radio is turned on.");
08     }
09     protected void turnOff() {
10        System.out.println("The radio is turned off.");
11     }
12     public void play(String music) {
13        System.out.println("The radio is playing "+ music + ".");
14     }
15     public void play(String singer, String music) {
16        System.out.println("The radio is playing "+ music + " of "+ singer + ".");
17     }
18     public static void furnish() {
19        System.out.println("This is a Radio.");
20     }
21  }
22  public class MethodTest {
23     public static void main(String[] args) throws InvocationTargetException,
24  IllegalAccessException {
25        Radio radio = new Radio();
26        Class c = radio.getClass();
27        Method[] methods = c.getDeclaredMethods();
28        System.out.println("共有方法："+ methods.length);
29        System.out.println("----------------");
30        for (Method m : methods) {
31           m.setAccessible(true);
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
32            System.out.println(Modifier.toString(m.getModifiers()) + " " +
33    m.getReturnType().getName() + " " + m.getName());
34          }
35          System.out.println("\n 调用结果：");
36          System.out.println("----------------");
37          for (Method m : methods) {
38             m.setAccessible(true);
39             String n = m.getName();
40             if (n.equalsIgnoreCase("play")) {
41                if (m.getParameterTypes().length == 1) {
42                   m.invoke(radio, "I love you more than I can say");
43                } else {
44                   m.invoke(radio, "Jerry Allison", "I love you more than I can
45    say");
46                }
47             } else {
48                m.invoke(radio);
49             }
50          }
51       }
52    }
```

【运行结果】程序运行结果如下。

```
共有方法：6
----------------
public void play
public void play
private java.lang.String brand
protected void turnOn
protected void turnOff
public static void furnish

每个方法的调用结果：
----------------
The radio is playing I love you more than I can say.
The radio is playing I love you more than I can say of Jerry Allison.
A famous brand: YongSheng
The radio is turned on.
The radio is turned off.
This is a Radio.
```

【代码解析】在以上程序中，利用反射列出了 Radio 的所有方法，包括相关的修饰符信息，然后调用方法。注意如何传递参数。



16.3　反射的安全性

通常，除了一个类内的代码，其他程序都不能访问这个类的私有变量和方法，但使用反射却可以绕过这个限制。反射中的 Field、Method 和 Constructor 类继承了 AccessibleObject 类，通过调用这三个类的 setAccessible (true)方法，就可以实现对这些属性或方法的调用。

有时候，这种操作会带来安全隐患，为此，需要启用 java.security.manager，检查程序是否具有调用 setAccessible()的权限。例如，当我们运行如下程序时，将抛出 java.security.AccessControlException。

```
>java -Djava.security.manager org.gaozou.book.javaabc.reflection.FieldTest
共有成员变量：5
------------------
Exception in thread "main" java.security.AccessControlException: access denied (
java.lang.reflect.ReflectPermission suppressAccessChecks)
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
        at java.security.AccessControlContext.checkPermission(Unknown Source)
        at java.security.AccessController.checkPermission(Unknown Source)
        at java.lang.SecurityManager.checkPermission(Unknown Source)
        at java.lang.reflect.AccessibleObject.setAccessible(Unknown Source)
        at org.gaozou.book.javaabc.reflection.FieldTest.main(FieldTest.java:18)
```

该程序不能执行，是因为默认安全策略禁止访问私有属性。通过创建一个包含如下规则的策略文件 policy.txt：

```
grant {
  permission java.security.AllPermission;
};
```

然后运行如下程序即可：

```
>java -Djava.security.policy
=policy.txt -Djava.security.manager org.gaozou.book.javaabc.reflection.FieldTest
共有成员变量：5
------------------
owner : George
space : 160
price : 300
address : Beijing
name : Happy House

为变量设置新值：
----------------
owner : George Li
space : 320
price : 600
address : Beijing China
name : Brilliant House
```

要创建策略文件，可以使用 policytool 工具，具体操作如下。

① 在命令行输入如下命令，打开“规则工具”软件，如图 16-1 所示。

```
>policytool
```

② 单击“添加规则项目”按钮，打开“规则项目”对话框，如图 16-2 所示。



图 16-1 规则工具

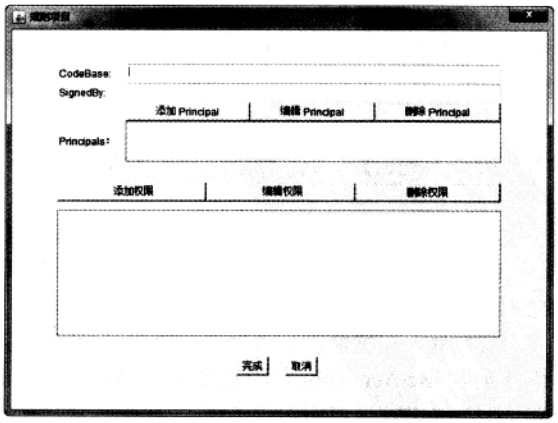


图 16-2 “规则项目”对话框

③ 单击“添加权限”按钮，打开“权限”对话框，如图 16-3 所示。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

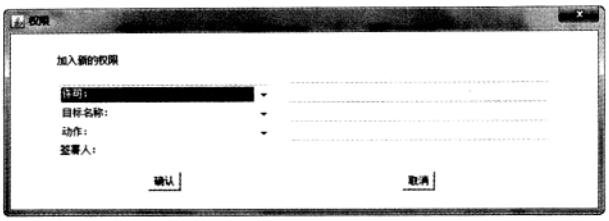


图 16-3　“权限”对话框

④ 在“许可”下拉列表中选择“AllPermission”选项，如图 16-4 所示，单击“确认”按钮，返回“规则项目”对话框。

⑤单击“完成”按钮，返回“规则工具”窗口。单击“文件”菜单项，选择“另存为”选项，如图 16-5 所示。

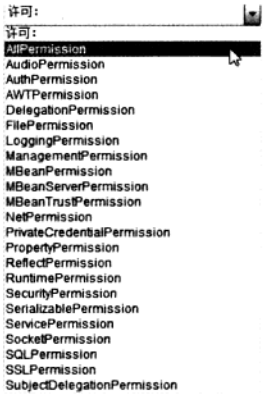


图 16-4　“许可”下拉列表



图 16-5　“文件”菜单

将此策略文件命名为“policy.txt”，保存在源代码包所在的目录下。

至此，就利用这个策略文件运行此目录下的所有程序。但要充分发挥 Java 安全管理器的功能，还要好好研究其提供的各种权限，运用各种权限，让程序能够安全灵活地运行。

16.4　习题

【习题 16-1】编写一个 Product 类，然后利用反射进行实例化，并且描述此类声明了多少属性和方法。

【分析】该题主要考查 Class 类的使用。

【核心代码】

```
public class Product {
   private String name;
   private String price;
   public Product() {
   }

   public Product(String name, String price) {
      this.name = name;
      this.price = price;
   }
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }

    public String getPrice() {
        return price;
    }
    public void setPrice(String price) {
        this.price = price;
    }

    private int cost() {
        return Integer.parseInt(price) / 2;
    }
}

public class DescribeClass {
    public static void main(String[] args) throws ClassNotFoundException,
IllegalAccessException, InstantiationException {
        Class c = Class.forName("org.gaozou.book.javaabc.exercises.ch16.Product");
        Product p = (Product) c.newInstance();
        System.out.println(c.getName() + " 有 "+ c.getDeclaredFields().length + " 个
属性");
        System.out.println(c.getName() + " 有 "+ c.getDeclaredMethods().length + "
个方法");
    }
}
```

【习题 16-2】利用反射描述第 1 题中 Product 类的属性，打印相应的描述信息。

【分析】该题主要考查 Field 类的使用。

【核心代码】

```
public class DescribeField {
    public static void main(String[] args) {
        Product p = new Product();
        Class c = p.getClass();
        Field[] fs = c.getDeclaredFields();
        System.out.println("Product 类有如下属性：");
        System.out.println("-----------------");
        for (Field f : fs) {
            System.out.println(Modifier.toString(f.getModifiers()) + " " + f.
getType().getName() + " " + f.getName());
        }
    }
}
```

【习题 16-3】利用反射描述第 1 题中 Product 类的方法，打印相应的描述信息。

【分析】该题主要考查 Method 类的使用。

【核心代码】

```
public class DescribeMethod {
    public static void main(String[] args) {
        Product p = new Product();
        Class c = p.getClass();
        Method[] ms = c.getDeclaredMethods();
        System.out.println("Product 类有如下方法：");
        System.out.println("-----------------");
        StringBuffer sb = new StringBuffer();
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
            for (Method m : ms) {
                sb.setLength(0);
                sb.append(Modifier.toString(m.getModifiers())).append("
").append( m.getReturnType().getName()).append(" ").append(m.getName()).append("(");
                Class[] ps = m.getParameterTypes();
                for (int i=0; i<ps.length; i++) {
                    sb.append(ps[i]);
                    if (i < ps.length - 1) {
                        sb.append(", ");
                    }
                }
                sb.append(")");
                System.out.println(sb);
            }
        }
    }
```

【习题 16-4】利用反射描述第 1 题中 Product 类的方法，打印相应的描述信息。

【分析】该题主要考查 Constructor 类的使用。

【核心代码】

```
public class DescribeConstructor {
    public static void main(String[] args) {
        Product p = new Product();
        Class c = p.getClass();
        Constructor[] cs = c.getDeclaredConstructors();
        System.out.println("Product 类有如下构造方法：");
        System.out.println("-----------------");
        StringBuffer sb = new StringBuffer();
        for (Constructor ct : cs) {
            sb.setLength(0);
            sb.append(Modifier.toString(ct.getModifiers())).append(" ").append(ct.
getName()).append("(");
            Class[] ps = ct.getParameterTypes();
            for (int i=0; i<ps.length; i++) {
                sb.append(ps[i]);
                if (i < ps.length - 1) {
                    sb.append(", ");
                }
            }
            sb.append(")");
            System.out.println(sb);
        }
    }
}
```

【习题 16-5】利用反射的方式对 Product 类进行实例化，明确反射的实例化机制。

【分析】该题主要考查 Constructor 类的使用。

【核心代码】

```
public class InstanceClass {
    public static void main(String[] args) throws ClassNotFoundException,
NoSuchMethodException, InvocationTargetException, IllegalAccessException,
InstantiationException {
        Class c = Class.forName("org.gaozou.book.javaabc.exercises.ch16.Product");
        Constructor c1 = c.getDeclaredConstructor();
        Product p1 = (Product) c1.newInstance();
        System.out.println("调用 p1 的 getClass()方法："+ p1.getClass());
        System.out.println("调用 p1 的 getName()方法："+ p1.getName());
        System.out.println("调用 p1 的 getPrice()方法："+ p1.getPrice());

        System.out.println("=========");
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
        Constructor c2 = c.getDeclaredConstructor(String.class, String.class);
        Product p2 = (Product) c2.newInstance("特步时尚板鞋", "200");
        System.out.println("调用 p2 的 getClass()方法："+ p2.getClass());
        System.out.println("调用 p2 的 getName()方法："+ p2.getName());
        System.out.println("调用 p2 的 getPrice()方法："+ p2.getPrice());
    }
}
```

【习题 16-6】利用反射调用 Person 类的私有和被保护属性，并给属性赋值。

【分析】该题主要考查 Class 及 Field 类的使用。

【核心代码】

```
public class Person {
    public String name;
    private String gender;
    protected int age;
    protected long money;
    private boolean married;

    public String speak() {
        return "speaking...";
    }

    protected String eat() {
        return "eating...";
    }

    public String eat(String food) {
        return "eat " + food;
    }
}

public class InvokeField {
    public static void main(String[] args) throws IllegalAccessException {
        Person p = new Person();
        Class c = p.getClass();
        Field[] fs = c.getDeclaredFields();
        for (Field f : fs) {
            f.setAccessible(true);
            System.out.println(f.getName() + "的值："+ f.get(p));
        }
        System.out.println("\n 为各个属性赋值后：");
        for (Field f : fs) {
            f.setAccessible(true);
            String n = f.getName();
            if (n.equalsIgnoreCase("name")) {
                f.set(p, "Jack");
            }
            if (n.equalsIgnoreCase("gender")) {
                f.set(p, "Male");
            }
            if (n.equalsIgnoreCase("age")) {
                f.set(p, 30);
            }
        }
        for (Field f : fs) {
            f.setAccessible(true);
            System.out.println(f.getName() + "的值："+ f.get(p));
        }
    }
}
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



【习题 16-7】利用反射调用类的私有和被保护的方法。

【分析】该题主要考查 Class 及 Method 类的使用。

【核心代码】

```
public class InvokeMethod {
    public static void main(String[] args) throws InvocationTargetException,
IllegalAccessException {
        Person p = new Person();
        Class c = p.getClass();
        Method[] ms = c.getDeclaredMethods();
        for (Method m : ms) {
            m.setAccessible(true);
            if (m.getName().equalsIgnoreCase("eat") && m.getParameterTypes().length
== 1) {
                System.out.println(m.invoke(p, "noodles"));
            } else {
                System.out.println(m.invoke(p));
            }
        }
    }
}
```

【习题 16-8】根据属性类型调用 Person 类的属性并赋值。

【分析】该题主要考查 Field 类的使用。

【核心代码】

```
public class InvokeFieldByType {
    public static void main(String[] args) throws IllegalAccessException {
        Person p = new Person();
        Class c = p.getClass();
        Field[] fs = c.getDeclaredFields();
        for (Field f : fs) {
            f.setAccessible(true);
            Class t = f.getType();
            if (t == String.class) {
                System.out.println(f.getName() + "的值："+ f.get(p));
            }
            if (t == int.class) {
                System.out.println(f.getName() + "的值："+ f.getInt(p));
            }
            if (t == long.class) {
                System.out.println(f.getName() + "的值："+ f.getLong(p));
            }
            if (t == boolean.class) {
                System.out.println(f.getName() + "的值："+ f.getBoolean(p));
            }
        }
        System.out.println("\n 为各个属性赋值后：");
        for (Field f : fs) {
            f.setAccessible(true);
            String n = f.getName();
            if (n.equalsIgnoreCase("name")) {
                f.set(p, "Jack");
            }
            if (n.equalsIgnoreCase("gender")) {
                f.set(p, "Male");
            }
            if (n.equalsIgnoreCase("age")) {
                f.setInt(p, 30);
            }
            if (n.equalsIgnoreCase("money")) {
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
                f.setLong(p, 10000000);
            }
            if (n.equalsIgnoreCase("married")) {
                f.setBoolean(p, false);
            }
        }
        for (Field f : fs) {
            f.setAccessible(true);
            System.out.println(f.getName() + "的值："+ f.get(p));
        }
    }
}
```

【习题 16-9】利用反射描述第 6 题中 Person 类的属性，打印相应的描述信息。

【分析】该题主要考查 Field 类及 Modifier 类的使用。

【核心代码】

```
public class DescribePersonField {
    public static void main(String[] args) {
        Person p = new Person();
        Class c = p.getClass();
        Field[] fs = c.getDeclaredFields();
        System.out.println("Person 类有如下属性：");
        System.out.println("-----------------");
        for (Field f : fs) {
            System.out.println(Modifier.toString(f.getModifiers()) + " " + f.
getType().getName() + " " + f.getName());
        }
    }
}
```

【习题 16-10】利用反射描述第 6 题中 Person 类的方法，打印相应的描述信息。

【分析】该题主要考查 Class 类及 Method 类的使用。

【核心代码】

```
public class DescribePersonMethod {
    public static void main(String[] args) {
        Person p = new Person();
        Class c = p.getClass();
        Method[] ms = c.getDeclaredMethods();
        System.out.println("Person 类有如下方法：");
        System.out.println("-----------------");
        StringBuffer sb = new StringBuffer();
        for (Method m : ms) {
            sb.setLength(0);
            sb.append(Modifier.toString(m.getModifiers())).append(" ").append( m.
getReturnType().getName()).append(" ").append(m.getName()).append("(");
            Class[] ps = m.getParameterTypes();
            for (int i=0; i<ps.length; i++) {
                sb.append(ps[i]);
                if (i < ps.length - 1) {
                    sb.append(", ");
                }
            }
            sb.append(")");
            System.out.println(sb);
        }
    }
}
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第 17 章　线　　程

线程是实现并发的一种有效的手段，多线程是现代操作系统开发的发展方向。这就好比一个人在手忙脚乱的时候，总希望再分身出一个自己，或者是恨不得多长出一只手来，那多出来的“分身”或者是希望多长出来的手就相当于线程，也就是多线程。本章主要介绍多线程的概念及使用。



17.1　线程的概念

在介绍线程之前，需要了解一下什么是进程，进程就是一个执行的程序任务，多进程的多任务处理特点就是允许计算机同时处理两个或两个以上的程序。线程则是比进程要小的执行单位，由前面介绍的进程来管理。一个多线程的程序要比多进程的程序需要更少的管理资源。

例如，需要造一辆汽车，首选需要制造汽车的外形，那就需要将一块块的钢板压制成形，然后将其焊接起来，接下来就是安装座椅、方向盘、发动机等各种零件，最后要送到喷漆车间上漆。这就是一条比较完整的汽车生产线，在各种零配件充足的情况下，它的生产量也是一定的。如果再建造一条这样的生产线，那样就会增加产量，两条生产线一起生产，就相当于双线程处理。

线程要比进程的划分粒度要小得多，如果需要新建一个进程，就要为其单独分配内存空间，而新建一个线程就不需要为其分配内存空间。所以，新建一个进程需要耗费更多时间和资源。线程不能够独立运行，必须驻留在一个进程内部，并且由进程为其实现任务的调度。



17.2　线程的创建

线程和普通的程序一样，都需要有第三方才可以启动。线程的创建包括有主线程的创建、实现 Runnable 接口和集成 Thread 类。在本节中就主要介绍线程的创建。

17.2.1　使用 Thread 创建线程

程序的主线程是在程序启动的时候就开始执行的，它是所有线程中最早开始运行的，也就是所有的 Java 程序在运行时都有一个线程，它还是产生其他子线程的线程，而且要用它来控制线程的各种动作。

主线程是在程序启动的时候自动创建的，但是它也是可以由 Thread 对象控制的，可以通过 Thread 对象调用相关的方法获取一个主线程的引用，然后控制其他线程。

【范例 17-1】示例代码 17-1 是一个使用 Thread 创建线程的示例。

示例代码 17-1

```
01  class MyThread extends Thread
02  {
03      //重写 run 方法
04      public void run() {
05          for(int i=0;i<10;i++)                    //设置 for 循环语句
```

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
06          {
07              try                                     //放入 try 语句块中
08              {
09                  Thread.sleep(100);                  //调用 sleep 方法让当前线程休眠
10              }
11              catch (InterruptedException e)          //catch 捕获程序异常
12              {
13                  e.printStackTrace();
14              }
15              System.out.println("第"+i+"MyThread");//在线程中打印出相关的信息
16          }
17      }
18  }
19  public class ThreadTest
20  {
21      public static void main(String[] args) {
22          MyThread mt=new MyThread();             //创建线程对象
23          mt.start();                             //启动线程
24          for(int i=0;i<10;i++)
25          {
26              try
27              {
28                  Thread.sleep(100);              //主线程休眠
29              }
30              catch (InterruptedException e)
31              {
32                  e.printStackTrace();
33              }
34              System.out.println("第"+i+"ThreadDemo");//主线程打印相关信息
35          }
36      }
37  }
```

【运行结果】程序运行结果如下。

```
第 0ThreadTest
第 0MyThread
第 1ThreadTest
第 1MyThread
第 2ThreadTest
第 2MyThread
第 3ThreadTest
第 3MyThread
第 4ThreadTest
第 4MyThread
第 5ThreadTest
第 5MyThread
第 6ThreadTest
第 6MyThread
第 7ThreadTest
第 7MyThread
第 8ThreadTest
第 8MyThread
第 9ThreadTest
第 9MyThread
```

【代码解析】在以上程序中，首先定义了一个线程类 MyThread，并重写了 run 方法（第 04～16 行）。在 run 方法中执行一个循环体，并且每次都调用 sleep 方法让当前线程休眠，并打印输出（第 05～15 行）。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



17.2.2　使用 Runnable 创建线程

前面使用的是继承 Thread 类的方法来创建的线程，在此使用 Runnable 接口的方法来创建线程。该方法是创建线程最简单的方法，只需要创建一个实现 Runnable 接口的类。Runnable 抽象了一个执行代码单元，可以通过实现 Runnable 接口来创建每一个对象的线程。

实现 Runnable 接口的类，需要定义一个名为 run()的无参方法。实现了 Runnable 接口的类不需要是 Thread 的子类就可以运行，只需要实例化一个 Thread 对象，并且将自己的引用作为一个参数传递给 Thread 的构造方法即可。

run()方法中可以定义创建一个新的线程，该方法在程序中确立另一个并发的线程执行入口。建立了新的线程之后，并不需要直接调 run()方法，而是由 start()方法执行对 run()方法的调用。

【范例 17-2】示例代码 17-2 是一个使用 Runnable 创建线程的示例。

示例代码 17-2

```
01  class MyThreadTest implements Runnable
02  {
03      //实现 run 方法
04      public void run()
05      {
06          for(int i=0;i<10;i++)                          //for 循环
07          {
08              try {
09                  Thread.sleep(100);                     //线程休眠
10              }
11              catch (InterruptedException e)             //捕获异常
12              {
13                  e.printStackTrace();
14              }
15              System.out.println("第"+i+"MyThreadTest");//线程信息
16          }
17      }
18  }
19  public class RunnableTest
20  {
21      public static void main(String[] args) {
22
23          MyThreadTest my=new MyThreadTest();            //创建 MyThread 类对象
24          Thread thr=new Thread(my);                     //构造新的线程对象
25          thr.start();                                   //启动线程
26          for(int i=0;i<10;i++)
27          {
28              try {
29                  Thread.sleep(100);                     //主线程休眠
30              }
31              catch (InterruptedException e)
32              {
33                  e.printStackTrace();
34              }
35              System.out.println("第"+i+"RunnableTest");//主线程信息
36          }
37      }
38  }
```

【运行结果】程序运行结果如下。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
第 0MyThreadTest
第 0RunnableTest
第 1MyThreadTest
第 1RunnableTest
第 2MyThreadTest
第 2RunnableTest
第 3RunnableTest
第 3MyThreadTest
第 4MyThreadTest
第 4RunnableTest
第 5RunnableTest
第 5MyThreadTest
第 6MyThreadTest
第 6RunnableTest
第 7RunnableTest
第 7MyThreadTest
第 8RunnableTest
第 8MyThreadTest
第 9MyThreadTest
第 9RunnableTest
```

17.2.3 两种方法比较

使用继承 Thread 和实现 Runnable 接口的方法来创建线程的区别在于创建线程的本质上。Thread 类提供有很多方法，只须重写 run()方法即可；Runnable 接口只有一个 run()方法需要实现。

通过实现 Runnable 接口来实现创建线程的方法要比使用 Thread 方法要好，而且体现了面向对象的编程思想。实现 Runnable 接口不影响继承其他类，也不影响实现其他的接口，只是让该类多了线程的能力，显然这样做使得程序更加灵活。

17.2.4 创建多线程

前面介绍了创建线程，本小节主要介绍创建多线程。

【范例 17-3】示例代码 17-3 是一个创建多线程的示例。

示例代码 17-3

```
01  public class MoreThreadTest {
02      public static void main(String[] args) {
03          Thread1 t1=new Thread1("frist");
04          Thread2 t2=new Thread2("second");
05          Thread3 t3=new Thread3("third");
06          t1.start();
07          t2.start();
08          t3.start();
09      }
10
11  }
12  class Thread1 extends Thread
13  {
14      String name;
15      Thread1(String threadname)
16      {
17          name=threadname;
18      }
19      public void run()
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
20      {
21          try{
22              for(int i=0;i<5;i++)
23              {
24                  System.out.println(name+": "+i);
25                  Thread.sleep(1000);
26              }
27          }catch(Exception e)
28          {
29              System.out.println(name+"异常");
30          }
31          System.out.println("退出");
32      }
33  }
34  class Thread2 extends Thread
35  {
36      String name;
37      Thread2(String threadname)
38      {
39          name=threadname;
40      }
41      public void run()
42      {
43          try{
44              for(int i=0;i<5;i++)
45              {
46                  System.out.println(name+": "+i);
47                  Thread.sleep(1000);
48              }
49          }catch(Exception e)
50          {
51              System.out.println(name+"异常");
52          }
53          System.out.println("退出");
54      }
55  }
56  class Thread3 extends Thread
57  {
58      String name;
59      Thread3(String threadname)
60      {
61          name=threadname;
62      }
63      public void run()
64      {
65          try{
66              for(int i=0;i<5;i++)
67              {
68                  System.out.println(name+": "+i);
69                  Thread.sleep(1000);
70              }
71          }catch(Exception e)
72          {
73              System.out.println(name+"异常");
74          }
75          System.out.println("退出");
76      }
77  }
```

【运行结果】程序运行结果如下。

```
frist: 0
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
second: 0
third: 0
frist: 1
third: 1
second: 1
frist: 2
second: 2
third: 2
frist: 3
second: 3
third: 3
frist: 4
second: 4
third: 4
退出
退出
退出
```

【代码解析】在以上程序中，分别创建了三个线程 Thread1、Thread2 和 Thread3（12、34 和 56 行），然后设置循环打印输出，并在打印完后休眠 1 秒（第 24～25 行），再在主方法中调用（第 03～08 行）。



17.3 线程的使用

17.3.1 开启线程

当线程创建完成后，它还只是一个普通的对象，并没有被真正地执行，如果要让它成为独立的执行线程，就必须要启动它。run()方法可以作为普通的方法一样调用，但是调用了 run()方法之后，并不代表新建了执行线程。要开启一个线程，只需要调用 start()方法就可以了。

【范例 17-4】示例代码 17-4 是一个开启线程的示例。

示例代码 17-4

```
01  public class MyRunnableTest {
02      public static void main(String[] args) {
03          //创建实现 Runnable 接口类的对象
04          RunnableTest rt=new RunnableTest();
05          //创建 Thread 对象，将第一步创建对象的引用作为构造器参数
06          Thread t=new Thread(rt);
07          //启动线程
08          t.start();
09      }
10  }
11  class RunnableTest implements Runnable
12  {
13      //实现 run 方法。
14      public void run() {
15          // TODO Auto-generated method stub
16          System.out.println("恭喜你，线程已经被启动" +
17                  "执行了 run 方法中的代码");
18      }
19  }
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



【运行结果】程序运行结果如下。

```
恭喜你，线程已经被启动执行了 run 方法中的代码
```

【代码解析】在以上程序中，首先创建 MyRunnableTest 类并实现 Runnable 接口（第 11 行）。重写 run 方法（第 14～18 行）。在主方法中调用 MyRunnableTest（第 03～08 行）。

17.3.2　线程休眠

休眠方法是让线程休眠一段指定的时间，等执行时间到了以后，线程会进入准备状态，等待线程调度程序的调度。线程休眠的方法有以下两个。

```
public static void sleep(long millis) throws InterruptedException
public static void sleep(long millis,int nanos) throws InterruptedException
```

以上两个休眠方法不同的是后一个方法更精确，它在指定的毫秒数加指定的纳秒数内让当前正在执行的线程休眠。

sleep()方法的使用格式如下。

```
try
        {
            //调用 sleep 方法让当前线程休眠
            Thread.sleep(100);
        }
        catch (InterruptedException e)
        {
            e.printStackTrace();
        }
```

【范例 17-5】示例代码 17-5 是一个线程休眠的示例。

示例代码 17-5

```
01  public class ThreadsleepTest
02  {
03      public static void main(String[] args) {
04          FirstTest one=new FirstTest();      //创建 FirstTest 对象
05          SecondTest two=new SecondTest();    //创建 secondTest 对象
06          ThirdTest threa=new ThirdTest();    //创建 thirdTest 对象
07          one.start();                        //开启 one 线程
08          two.start();                        //开启 two 线程
09          threa.start();                      //开启 threa 线程
10      }
11  }
12  //创建第一个线程
13  class FirstTest extends Thread{
14          public void run() {                 //   实现 run()方法。
15                                              // TODO 自动生成方法存根
16              try{
17              for(int a=0;a<5;){              //循环打印数字
18                  a++;                        //a 增加 1
19                  System.out.println("第一个线程   第"+a+"次循环");//打印输出
20                  Thread.sleep(2000);         //间隔 2 秒打印一次
21              }
22              }catch(Exception e){
23                  System.out.println("异常");
24              }
25          }
26  }
27  //创建第二个线程
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
28  class SecondTest extends Thread{
29          public void run() {                    //实现 run()方法。
30                                                 // TODO 自动生成方法存根
31              try{
32                  for(int a=0;a<5;){             //循环打印数字
33                      a++;
34                  System.out.println("第二个线程   第"+a+"次循环");//打印输出
35                      Thread.sleep(2500);            //间隔 2.5 秒打印一次
36                  }
37                  }catch(Exception e){
38                      System.out.println("异常");    //打印输出异常
39                  }
40          }
41  }
42  //创建第三个线程
43  class ThirdTest extends Thread{
44          public void run() {                        //实现 run()方法。
45              // TODO 自动生成方法存根
46              try{
47                  for(int a=0;a<5;){                 //循环打印数字
48                      a++;
49              System.out.println("第三个线程   第"+a+"次循环");//打印输出
50                      Thread.sleep(3000);            //间隔 3 秒打印一次
51                  }
52                  }catch(Exception e){
53                      System.out.println("异常");    //打印输出
54                  }
55          }
56      }
```

【运行结果】程序运行结果如下。

```
第一个线程   第 1 次循环
第三个线程   第 1 次循环
第二个线程   第 1 次循环
第一个线程   第 2 次循环
第二个线程   第 2 次循环
第三个线程   第 2 次循环
第一个线程   第 3 次循环
第二个线程   第 3 次循环
第三个线程   第 3 次循环
第一个线程   第 4 次循环
第二个线程   第 4 次循环
第一个线程   第 5 次循环
第三个线程   第 4 次循环
第二个线程   第 5 次循环
第三个线程   第 5 次循环
```

【代码解析】在以上程序中，对于三个线程，分别设置了三个不同的线程休眠时间（20、35、50 行），在第四次循环时，就开始出现交错显示了。

在程序中使用到了线程构造器，线程中的构造器主要包括：

- Thread()
- Thread(Runnable target)
- Thread(Runnable target,String name)
- Thread(String name)
- Thread(ThreadGroup group,Runnable target)

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



- Thread(ThreadGroup group,Runnable target,String name)
- Thread(ThreadGroup group,String name)

创建多线程时，使用不同的构造器，其运行结果有可能会不同，不过每个线程都会正常运行。

17.3.3　等待线程

汽车在行驶在马路上时，遇到红灯总会停下来，等绿灯之后才可以继续行驶。在线程中，也可以让线程暂时停止下来。join()方法又叫加入方法，就是暂停当前线程的执行，直到调用join()方法的线程执行完毕，才继续执行。

【范例 17-6】示例代码 17-6 是一个等待线程的示例。

示例代码 17-6

```
01  public class ThreadjoinTest
02  {
03      public static void main(String[] args) {
04          FirstTest1 one=new FirstTest1();          //创建 FirstTest 对象
05          SecondTest1 two=new SecondTest1();        //创建 secondTest 对象
06          ThirdTest1  threa=new ThirdTest1();       //创建 thirdTest 对象
07          one.start();                              //开启 one 线程
08          two.start();                              //开启 two 线程
09          threa.start();                            //开启 threa 线程
10      }
11  }
12  //创建第一个线程
13  class FirstTest1 extends Thread{
14  //   实现 run()方法。
15          public void run() {
16              // TODO 自动生成方法存根
17              try{
18              for(int a=0;a<10;){                   //循环打印数字
19                  a++;
20                  System.out.println("第一个线程   第"+a+"次循环");    //打印输出
21                  Thread.sleep(1000);               //间隔 1 秒打印一次
22              }
23              }catch(Exception e){
24                  System.out.println("异常");       //打印输出异常
25              }finally{
26              }
27          }
28  }
29  //创建第二个线程
30  class SecondTest1 extends Thread{
31      public FirstTest1 first;
32          public void run() {                       //实现 run()方法。
33              // TODO 自动生成方法存根
34                  for(int a=0;a<10;){               //循环打印数字
35                      try{
36                      a++;
37                  System.out.println("第二个线程   第"+a+"次循环");//打印输出
38                      Thread.sleep(1000);           //间隔 1 秒打印一次
39                  }catch(Exception e){
40                      System.out.println("异常");//打印输出异常
41                  }
42                  if(a==4){
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
43                          try {
44                              first.join();          //加入线程 FirstTest1
45                          } catch (InterruptedException e) {
46                              // TODO 自动生成 catch 块
47                              e.printStackTrace();
48                          }
49                      }
50                      }
51              }
52      }
53      //创建第三个线程
54      class ThirdTest1 extends Thread{
55      //   实现 run()方法。
56              public void run() {
57                  // TODO 自动生成方法存根
58                  try{
59                      for(int a=0;a<10;){                          //循环打印数字
60                          a++;
61                  System.out.println("第三个线程   第"+a+"次循环");//打印输出
62                          Thread.sleep(1000);                      //间隔 1 秒打印一次
63                      }
64                      }catch(Exception e){
65                          System.out.println("异常");          //打印输出异常
66                      }
67              }
68      }
```

【运行结果】程序运行结果如下。

```
第二个线程   第 1 次循环
第一个线程   第 1 次循环
第三个线程   第 1 次循环
第一个线程   第 2 次循环
第三个线程   第 2 次循环
第二个线程   第 2 次循环
第一个线程   第 3 次循环
第三个线程   第 3 次循环
第二个线程   第 3 次循环
第一个线程   第 4 次循环
第二个线程   第 4 次循环
第三个线程   第 4 次循环
第三个线程   第 5 次循环
第一个线程   第 5 次循环
第三个线程   第 6 次循环
第一个线程   第 6 次循环
第三个线程   第 7 次循环
第一个线程   第 7 次循环
第三个线程   第 8 次循环
第一个线程   第 8 次循环
第三个线程   第 9 次循环
第一个线程   第 9 次循环
第三个线程   第 10 次循环
第一个线程   第 10 次循环
```

【代码解析】在程序中，分别将三个线程的休眠时间改为 1 秒（21、38、62 行），在第二个线程中，循环到第五次的时候，加入线程 1 的 join()方法（第 42～48 行）。在第六次以后，第一个线程和第三个线程交替运行，直到结束。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



17.4　线程的控制

17.4.1　线程的优先级

线程的优先级用来判定何时运行该线程。优先级高的线程能够比优先级低的线程获得更多的 CPU 时间，设置线程的优先级，可以使用 setPriority()方法来进行，使用方法如下。

```
final void setPriority(int level)
```

其中的整型参数 level，用来设置优先级的大小，优先级最小的表示方法为 MIN_PRIORITY，优先级最大的表示方法为 MAX_PRIORITY，level 的值在 1～10 之间，也就是优先级最小的 level 的值为 1，优先级最大的 level 的值为 10。

获得当前线程优先级的方法为 getPriority()。

【范例 17-7】示例代码 17-7 是一个测试线程优先级的示例。

示例代码 17-7

```
01  public class ThreadPriority {
02      public static void main(String[] args) { // java 程序主入口处
03          List list = new ArrayList();          // 创建一个 List 对象
04          Student stu1 = new Student("张三", 5, 98, 130);// 创建一个 Student 对象
05          list.add(stu1);                       //将 Student 对象添加到 List 列表中
06          Student stu2 = new Student("李四", 3, 95, 150);
07          list.add(stu2);
08          Student stu3 = new Student("王五", 4, 100, 120);
09          list.add(stu3);
10          Student stu4 = new Student("赵六", 6, 91, 140);
11          list.add(stu4);
12          Student stu5 = new Student("钱七", 7, 93, 150);
13          list.add(stu5);
14          System.out.println("按成绩分数的高低排坐");
15          new Score().RowSeat(list);
16          System.out.println("按个子的高低排坐");
17          new Height().RowSeat(list);
18      }
19  }
20  class Height implements Runnable {              // 线程实现接口 Runnable
21      Thread thread;                              // 声明一个线程
22      Height() {
23      }                                           // 默认构造方法
24      Height(String name) {                       // 构造方法初始一个线程
25          thread = new Thread(this, name);
26      }
27      public void run() {
28      // 获得线程的名称
29      System.out.println("一年一班的学生成员：" + thread.getName());
30      }
31      public void RowSeat(List list) {            // 按优先级执行线程
32          Student st = (Student) list.get(0);  // 张三
33          Student st1 = (Student) list.get(1);
34          Student st2 = (Student) list.get(2);
35          Student st3 = (Student) list.get(3);
36          Student st4 = (Student) list.get(4);
37          Height o1 = new Height(st.name);      // 创建线程对象并命名
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
38          Height o2 = new Height(st1.name);
39          Height o3 = new Height(st2.name);
40          Height o4 = new Height(st3.name);
41          Height o5 = new Height(st4.name);
42          o1.thread.setPriority(Thread.NORM_PRIORITY - 1);     // 较次之
43          o2.thread.setPriority(Thread.MAX_PRIORITY);//MAX_PRIORITY 优先级最大
44          o3.thread.setPriority(Thread.MIN_PRIORITY);//MIN_PRIORITY 优先级最小
45      o4.thread.setPriority(Thread.NORM_PRIORITY);// NORM_PRIORITY 处于中间位置
46          o5.thread.setPriority(Thread.MAX_PRIORITY - 1);// 比中间位置的小一些
47          o1.thread.start();                          // 启动线程
48          o2.thread.start();
49          o3.thread.start();
50          o4.thread.start();
51          o5.thread.start();
52          try {
53              o5.thread.join();                       // 等待线程运行结束
54          } catch (InterruptedException e) {          // 捕获拦截异常
55              System.out.println("等待线程结束出错：" + e.getMessage());
56          }
57      }
58  }
59  class Score extends Thread {
60      Thread thread;
61      public Score() {
62      }                                               // 默认构造方法
63      public Score(String name) {                     // 带参数构造方法初始一个线程
64          thread = new Thread(this, name);
65      }
66      public void run() {
67      // 获得线程的名称
68          System.out.println("一年一班的学生成员：" + thread.getName());
69      }
70      public void RowSeat(List list) {                // 按优先级执行线程
71          Student st = (Student) list.get(0);
72          Student st1 = (Student) list.get(1);
73          Student st2 = (Student) list.get(2);
74          Student st3 = (Student) list.get(3);
75          Student st4 = (Student) list.get(4);
76          Score e1 = new Score(st.name);              // 创建线程对象并命名
77          Score e2 = new Score(st1.name);
78          Score e3 = new Score(st2.name);
79          Score e4 = new Score(st3.name);
80          Score e5 = new Score(st4.name);
81          // 设置优先级 10 的优先级最高，1 的最小
82          e1.thread.setPriority(st.score / 10);
83          e2.thread.setPriority(st1.score / 10);
84          e3.thread.setPriority(st2.score / 10);
85          e4.thread.setPriority(st3.score / 10);
86          e5.thread.setPriority(st4.score / 10);
87          e1.thread.start();                          // 启动线程
88          e2.thread.start();
89          e3.thread.start();
90          e4.thread.start();
91          e5.thread.start();
92          try {
93              e5.thread.join();                       // 等待线程运行结束
94          } catch (InterruptedException e) {          // 捕获拦截异常
95              System.out.println("等待线程结束出错：" + e.getMessage());
96          }
97      }
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
98  }
99  class Student {                                    // 学生类
100     public String name;                            // 姓名
101     public int age;                                // 年龄
102     public int score;                              // 成绩
103     public int height;                             // 身高
104     // 构造函数为各成员变量
105     public Student(String name, int age, int score, int height) {
106         this.name = name;
107         this.age = age;
108         this.height = height;
109         this.score = score;
110     }
111 }
```

【运行结果】程序运行结果如下。

```
按成绩分数的高低排坐
一年一班的学生成员：张三
一年一班的学生成员：李四
一年一班的学生成员：王五
一年一班的学生成员：赵六
一年一班的学生成员：钱七
按个子的高低排坐
一年一班的学生成员：李四
一年一班的学生成员：张三
一年一班的学生成员：赵六
一年一班的学生成员：钱七
一年一班的学生成员：王五
```

17.4.2 线程让步

行驶在公路上，如果遇到有鸣笛急救车或者是鸣笛警车，要主动为他们让路。同样，线程中也有让步方法，yield()方法又叫让步，可以使当前运行的线程回到可运行的状态，让具有相同优先级的线程获得运行的机会。虽然称 yield 方法为让步方法，但其往往达不到让步的效果，它只是让当前的线程回到可以运行的状态。

【范例 17-8】示例代码 17-8 是一个测试线程让步的示例。

示例代码 17-8

```
01  public class ThreadYieldTest
02  {
03      public static void main(String[] args) {
04          FirstTest2 one=new FirstTest2();          //创建 FirstTest 对象
05          SecondTest2 two=new SecondTest2();        //创建 secondTest 对象
06          ThirdTest2  threa=new ThirdTest2();       //创建 thirdTest 对象
07          one.start();                              //开启 one 线程
08          two.start();                              //开启 two 线程
09          threa.start();                            //开启 threa 线程
10      }
11      }
12  //创建第一个线程
13  class FirstTest2 extends Thread
14  {
15  //   实现 run()方法。
16          public void run()
17          {
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
18                  // TODO 自动生成方法存根
19                  try{
20                  for(int a=0;a<5;){                    //循环打印数字
21                      a++;
22                      System.out.println("第一个线程   第"+a+"次循环");//打印输出
23                  }
24                  }catch(Exception e){
25                      System.out.println("异常");      //输出异常
26                  }finally{
27                  }
28
29              }
30
31  }
32  //创建第二个线程
33  class SecondTest2 extends Thread
34  {
35  //   实现 run()方法。
36          public void run()
37          {
38              // TODO 自动生成方法存根
39              try{
40                  for(int a=0;a<5;)
41                  {                                                    //循环打印数字
42                      a++;
43              System.out.println("第二个线程   第"+a+"次循环");    //打印输出
44                      Thread.yield();                                  //线程让步
45                  }
46                  }catch(Exception e){
47                      System.out.println("异常");                   //打印输出异常
48                  }
49          }
50  }
51  //创建第三个线程
52  class ThirdTest2 extends Thread
53  {
54  //   实现 run()方法。
55          public void run()
56          {
57              // TODO 自动生成方法存根
58              try{
59                  for(int a=0;a<5;)
60                  {                                                    //循环打印数字
61                      a++;
62              System.out.println("第三个线程   第"+a+"次循环");    //打印输出
63              }
64                  }catch(Exception e)
65                  {
66                      System.out.println("异常");                   //打印输出异常
67                  }
68          }
69      }
```

【运行结果】程序运行结果如下。

```
第一个线程   第 1 次循环
第二个线程   第 1 次循环
第三个线程   第 1 次循环
第一个线程   第 2 次循环
第三个线程   第 2 次循环
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
第一个线程　第 3 次循环
第三个线程　第 3 次循环
第一个线程　第 4 次循环
第一个线程　第 5 次循环
第二个线程　第 2 次循环
第三个线程　第 4 次循环
第三个线程　第 5 次循环
第二个线程　第 3 次循环
第二个线程　第 4 次循环
第二个线程　第 5 次循环
```

【代码解析】在以上程序中，在第二个线程中加入 yield()方法（第 44 行）。运行时，先运行了四次第一个线程，第三个线程和第二个线程交替运行了一次，后面开始运行第三个线程，第三个线程结束后，运行第二个线程，一直到程序结束。

17.4.3　线程同步

为了便于理解同步问题，这里用一个实例程序来说明同步问题的由来。假如在一间办公室里只有一台打印机，这台打印机在收到信息后就会把信息打印出来。某天，有多个老师要使用它打印学生成绩，他们可能同时进行请求，即在打印机处理一个打印任务的时候，又同时收到了其他的打印请求。老师们希望打印机可以一个个地处理打印任务，但在这种情况下显然会很容易出现混乱。

【范例 17-9】示例代码 17-9 是一个普通打印的示例。

示例代码 17-9

```
01  //Printer 类用来模拟打印机，它可以获得信息并把它打印出来
02  class Printer
03  {
04      //打印方法
05      void printScore(String name,int score1,int score2,int score3){
06          System.out.println(name+"的成绩：");
07          System.out.print(name+"英语：");
08          System.out.println(score1);
09          //为了是程序运行结果明显，在打印的过程中让打印工作休眠一段时间
10          try {
11              Thread.sleep(1000);
12          }
13   catch (InterruptedException e)
14   {
15              e.printStackTrace();
16          }
17          System.out.print(name+"数学：");
18          System.out.println(score2);
19          System.out.print(name+"语文：");
20          System.out.println(score3);
21      }
22  }
23  //教师类，每个教师都有自己的线程并且公用一个打印机
24  class Teacher implements Runnable{
25      Printer ps=null;                          //打印机
26      String name=null;                         //姓名
27      Thread t;                                 //线程
28      int score1=0;                             //成绩
29      int score2=0;
30      int score3=0;
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
31        //教师类的构造函数
32        Teacher(Printer printer,String nm,int s1,int s2,int s3){
33            ps=printer;
34            name=nm;
35            t=new Thread(this);
36            score1=s1;
37            score2=s2;
38            score3=s3;
39        }
40        //教师调用打印机
41        public void run() {
42            ps.printScore(name, score1, score2, score3);
43        }
44    }
45    public class SynchDemo {
46        public static void main(String[] args) {
47            //创建一个打印机对象
48            Printer ps=new Printer();
49            //创建 3 个教师实例
50            Teacher t1=new Teacher(ps,"lancy",87,76,90);
51            Teacher t2=new Teacher(ps,"macula",97,86,70);
52            Teacher t3=new Teacher(ps,"herbert",67,86,97);
53            //启动打印工作
54            t1.t.start();
55            t2.t.start();
56            t3.t.start();
57        }
58    }
```

【运行结果】程序的运行结果如下。

```
lancy 的成绩:
lancy 英语: 87
macula 的成绩:
macula 英语: 97
herbert 的成绩:
herbert 英语: 67
lancy 数学: 76
lancy 语文: 90
macula 数学: 86
macula 语文: 70
herbert 数学: 86
herbert 语文: 97
```

【代码解析】Printer 类是打印机类，它有一个 printScore 方法，可以把接收到的信息打印出来，为了使运行结果明显，程序在打印过程中让打印工作有一段时间的暂停（第 02～22 行）。Teacher 类是教师类，实现了 Runnable 接口，它是一个单独的线程可以随时进行打印，并且由它来提供打印信息（第 24～44 行）。在类 SynchDemo 中，创建了一个打印机实例，有三个老师提交了打印请求，打印请求虽然提交的时间不同，但是会在前面的打印完成之前进行提交（第 45～58 行）。可以看到，打印的结果并不是一个个地把任务完成后再去执行其他的任务，因此这显然不能满足程序的要求。

在 Java 中实现同步有两种方式，即同步方法和同步块。同步方法的使用很简单，只须在需要同步的方法声明的时候加上 synchronized 关键字即可，它的格式如下：

```
synchronized 方法声明
{
  //方法体
}
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

同步块的格式如下：

```
synchronized（资源对象）
{
  //需要进行同步的方法
}
```

下面使用这两种方式解决前面一小节的问题。实现很简单，只须在打印机类 Printer 的 printScore 方法前加上关键字 synchronized 即可，方法头声明如下：

```
synchronized  void printScore(String name,int score1,int score2,int score3)
{
//方法体不变
}
```

重新编译运行程序，可以得到如下结果：

```
lancy 的成绩：
lancy 英语：87
lancy 数学：76
lancy 语文：90
herbert 的成绩：
herbert 英语：67
herbert 数学：86
herbert 语文：97
macula 的成绩：
macula 英语：97
macula 数学：86
macula 语文：70
```

显然这才是想要的结果。

下面的代码是使用同步块来解决这个问题的。需要改动的是教师类 Teacher 中的 run 方法，改动程序如下：

```
public void run()
{
     synchronized(ps)       //锁定 ps 对象，让其单线程操作
    {
     ps.printScore(name, score1, score2, score3);     //调用 printScore 方法
    }
}
```

运行程序可以得到相同的结果。

Java 的同步是通过锁的机制来实现的，当线程进入同步方法的时候，该线程将会获得同步方法所属对象的锁，一旦获得对象锁，则其他线程不能再执行被锁对象的其他任何同步方法。只有在同步方法执行完毕释放锁之后，才能执行。下面的程序说明了这一点。

【范例 17-10】示例代码 17-10 是一个使用线程同步实现打印的示例。

示例代码 17-10

```
01  class Source
02  {
03        //定义同步方法 method1
04        synchronized void method1(){
05           System.out.println("进入方法 method1，获得锁");
06          try {
07                Thread.sleep(1000);                  //线程休眠
08           }
09   catch (InterruptedException e)
10   {
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
11                e.printStackTrace();
12            }
13           System.out.println("mehtod1 执行完毕，释放锁");
14       }
15       //定义同步方法 method2
16       synchronized void method2(){
17            System.out.println("进入方法 method2，获得锁");
18           try {
19                Thread.sleep(1000);                    //线程休眠
20            }
21   catch (InterruptedException e)
22    {
23                e.printStackTrace();
24            }
25           System.out.println("mehtod2 执行完毕，释放锁");
26       }
27   }
28   //定义线程类
29   class MyThread implements Runnable{
30       String name;                                    //通过 name 来判断调用哪个方法
31       Source s=null;
32       MyThread(Source s,String nm)                    //构造方法
33   {
34            this.s=s;
35            name=nm;
36       }
37       public void run() {                             //线程执行时的操作
38            if("method1".equals(name))
39                s.method1();                           //调用同步方法 1
40            else
41            {
42                System.out.println("Thread2 启动，准备调用 method2");
43                s.method2();                           //调用同步方法 2
44            }
45       }
46   }
47   public class SynchMethodDemo {
48       public static void main(String[] args)
49   {
50            Source s=new Source();                     //创建一个资源对象
51            //创建两个实现 Runnable 接口的对象
52            MyThread mt1=new MyThread(s,"method1");
53            MyThread mt2=new MyThread(s,"method2");
54            //创建两个线程
55            Thread t1=new Thread(mt1);
56            Thread t2=new Thread(mt2);
57            //启动线程
58            t1.start();
59            t2.start();
60       }
61   }
```

【运行结果】程序运行结果如下。

```
进入方法 method1，获得锁
Thread2 启动，准备调用 method2
mehtod1 执行完毕，释放锁
进入方法 method2，获得锁
mehtod2 执行完毕，释放锁
```

【代码解析】程序首先在类 Source 中定义了两个同步方法（第 01～08 行）。在线程类

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

MyThread 启动的时候，根据传递给它的参数 name 来判断调用哪个方法（第 29～46 行）。在程序入口 SynchMethodDemo 中定义了两个线程，它们会调用不同的方法，启动两个线程（第 47～59 行）。

可以看到程序运行的结果。由于两个方法都是同步方法，在线程 Thread1 调用方法 method1 之后，即使线程 Thread2 调用同步方法 method2 也不能执行，而必须等待 method1 执行完毕之后才能获得锁，从而执行同步方法 method2。

17.4.4　多线程的死锁问题

在任何的并发运行的系统中，都存在有多个访问者资源互斥时就会发生死锁的现象。如果系统的设计不完善，在 Java 中进行多线程的开发就可能会发生死锁。Java 中的死锁就是线程之间相互等待对方释放资源对象的锁，而每一个线程又持有其他线程所需要的锁，这就造成了几个线程处于没有必要的永久等待中。

【范例 17-11】示例代码 17-11 是一个多线程死锁的示例。

示例代码 17-11

```
01  class Resource                              //资源类
02  {
03      String resoureName;                     //资源名字
04      Resource(String name) {
05          this.resoureName = name;
06      }
07  }
08  class MyThread implements Runnable          //实现 Runnble 接口
09  {
10      Resource rs1;                           //线程需要的资源 1
11      Resource rs2;                           //线程需要的资源 1
12      String name;
13      MyThread(Resource rs1, Resource rs2, String name) {//构造方法
14          this.rs1 = rs1;
15          this.rs2 = rs2;
16          this.name = name;
17      }
18      //run 方法有两个同步块语句，只有它同时占有两个资源的锁时才能执行完毕
19      public void run() {
20          synchronized (rs1)                  //对资源 1 同步锁定
21          {
22              System.out.println(this.name + "获得  " + rs1.resoureName);
23              try
24              {
25                  Thread.sleep(1000);   //线程休眠
26              }
27              catch (InterruptedException e)
28              {
29                  e.printStackTrace();
30              }
31          System.out.println(this.name+"等待"+rs2.resoureName+"释放");
32          synchronized (rs2)                  //对资源 2 同步锁定
33          {
34              System.out.println(this.name + "获得  " + rs2.resoureName);
35              try
36              {
37                  Thread.sleep(1000);   //线程休眠
38              }
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
39                catch (InterruptedException e)
40                {
41                    e.printStackTrace();
42                }
43            }
44            }
45        }
46    }
47    public class DeadlockDemo {
48        public static void main(String[] args) {
49            //创建资源对象
50            Resource rs1=new Resource("资源 1");
51            Resource rs2=new Resource("资源 2");
52            Resource rs3=new Resource("资源 3");
53            //创建三个实现 Runnable 接口的对象
54            MyThread ms1=new MyThread(rs1,rs2,"线程 1");
55            MyThread ms2=new MyThread(rs2,rs3,"线程 2");
56            MyThread ms3=new MyThread(rs3,rs1,"线程 3");
57            //创建线程
58            Thread t1=new Thread(ms1);
59            Thread t2=new Thread(ms2);
60            Thread t3=new Thread(ms3);
61            //启动线程
62            t1.start();
63            t2.start();
64            t3.start();
65        }
66    }
```

【运行结果】程序运行结果如下。

```
线程 1 获得  资源 1
线程 3 获得  资源 3
线程 2 获得  资源 2
线程 1 等待资源 2 释放
线程 3 等待资源 1 释放
线程 2 等待资源 3 释放
```

【代码解析】程序中 Resource 类代表了资源类（第 01～07 行），MyThread 类是创建线程的类。在该线程中需要有两个资源才能正常运行完程序。但是在运行程序的时候，ms1 占有了资源 1，ms2 占有了资源 2，ms3 占有了资源 3（第 19～43 行），它们都在等待对方释放资源，因而都不能执行。

17.5 线程池

如果要启动一个线程来完成指定的任务，首先需要创建线程对象，而且要在任务执行完毕之后使线程死亡。这种模式在一定程度上会影响程序的执行性能，因为创建线程对象和清除线程垃圾会占用大量的 CPU 等系统资源，此时使用线程池就可以很好地解决资源浪费的问题。

线程池就是在系统中专门开辟一块区域，其中存放一些待命的线程，这些线程就组成了线程池。例如，酒店一般都有自己的用来清洗被褥和床单的“洗刷厂”，这里面有一些洗衣机，可以将存放洗衣机的房间看做是一个线程池，因洗衣机容量有限，所以会将需要清洗的被褥分批次放入不同的洗衣机，而且会根据数量的不同依次增加或者减少使用洗衣机的数量。

实际的开发中，线程池的类型也不同，常用的有固定尺寸的线程池和可变尺寸的线程池。固定尺寸的线程池中，待命线程的数量是一定的，并且实现起来比较简单，不过，如果需要执

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

行的任务过多，就可能会造成大量的等待；如果执行任务过少，也会造成资源的浪费。

可变尺寸的线程池中待命的线程数量是根据任务负载的需要来进行动态变化的，即需要执行任务如果比较多，就会动态增加线程的数量，提高线程的效率；如果任务比较少，则系统就会自动缩减待命线程的数量，用以减少内存的消耗。不过这种可变尺寸的线程池实现起来比较复杂。

【范例 17-12】示例代码 17-12 是一个线程池示例。

示例代码 17-12

```
01  public class SurgeryThreadPool {
02      private static Runnable createTask(final int taskID) {// 创建任务方法
03          return new Runnable() {
04              public void run() {              // 创建手术任务
05                  System.out.println("手术开始,编号为" + taskID);
06                  System.out.println("正在手术中～～");
07                  System.out.println("手术结束,编号为" + taskID);
08              }
09          };
10      }
11      public static void main(String[] args) { // java 程序主入口处
12      // 创建有 3 个任务线程的一个线程池
13          ThreadTask threadPool = new ThreadTask(3);
14          try {                               // 休眠 600ms,让线程池中的任务线程全部运行
15              Thread.sleep(600);
16          } catch (InterruptedException e) { // 捕获拦截异常
17              System.out.println("线程休眠出错: " + e.getMessage());
18          }
19          for (int i = 0; i < 3; i++) { // 循环创建并执行任务
20              threadPool.addTask(createTask(i));
21          }
22          threadPool.waitTaskColsed();   // 等待所有的任务执行完毕
23          threadPool.closePool();        // 关闭线程池
24      }
25  }
26  class ThreadTask extends ThreadGroup { // 继承线程组实现线程池功能
27      private boolean isStop = false;    // 线程池是否关闭
28      private LinkedList queue;          // 工作任务队列
29      private static int poolID = 1;     // 线程池的编号
30      // 负责从工作队列中取出任务并执行的内部类
31  private class SurgeryTask extends Thread {
32          private int id;                // 任务编号
33          public SurgeryTask(int id) {   // 构造方法进行初始化
34              super(ThreadTask.this, id + ""); // 将线程加入当前线程组中
35              this.id = id;
36          }
37          public void run() {
38              while (!isInterrupted()) {// 判断线程是否被中断
39                  Runnable task = null;
40                  task = getTask(id);   // 取出任务
41              // 如果 getTask()返回 null 或者线程执行 getTask()时被中断，则结束此线程
42                  if (task == null)
43                      return;
44                  try {
45                      task.run();       // 运行任务
46                  } catch (Throwable t) {
47                      t.printStackTrace();
48                  }
49              }
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
50          }
51      }
52      public ThreadTask(int size) {       // 构造方法传入线程池中的工作线程的数量
53          super(poolID + "");             // 指定线程组名称
54          setDaemon(true);                // 继承线程组的方法用来设置是否守护线程池
55          queue = new LinkedList();       // 创建工作任务队列
56          for (int i = 0; i < size; i++) {// 循环创建任务线程
57              new SurgeryTask(i).start(); // 根据线程池数据创建任务线程并启动线程,
58          }
59      }
60      public synchronized void addTask(Runnable task) {// 添加新任务并执行任务
61          if (isStop) {                               // 判断标识
62              throw new IllegalStateException(); // 抛出不合理的状态异常
63          }
64          if (task != null) {
65              queue.add(task);                // 向任务队列中加入一个任务
66              notify();                       // 唤醒等待任务的工作任务线程
67          }
68      }
69      private synchronized Runnable getTask(int id) { // 取出任务
70          try {
71              while (queue.size() == 0) {     // 循环使线程等待任务
72                  if (isStop)
73                      return null;
74                  System.out.println("病人" + id + "正在等待手术...");
75                  wait();                     // 如果任务队列中没有任务,就等待任务
76              }
77          } catch (InterruptedException e) { // 捕获拦截异常
78              System.out.println("等待治疗出现错误: " + e.getMessage());
79          }
80          System.out.println("病人" + id + "开始执行手术...");
81          return (Runnable) queue.removeFirst(); // 返回第一个任务并从队列中删除
82      }
83      public synchronized void closePool() { // 关闭线程池
84          if (!isStop) {                      // 判断标识
85              waitTaskColsed();               // 等待任务线程执行完毕
86              isStop = true;                  // 标识为真
87              queue.clear();                  // 任务队列清空
88              interrupt();                    // 唤醒线程池中所有的工作线程
89          }
90      }
91      public void waitTaskColsed() {          // 等待任务线程把所有的任务执行完毕
92          synchronized (this) {
93              isStop = true;                  // 标识为真
94              notifyAll();                    // 唤醒等待任务的工作任务线程
95          }
96          Thread[] threads = new Thread[activeCount()];// 创建线程组中活动的线程组
97          int count = enumerate(threads);   // 获得线程组中当前所有的活动的工作线程
98          for (int i = 0; i < count; i++) {// 循环等待所有的工作线程结束
99              try {
100                 threads[i].join();              // 等待工作线程结束
101             } catch (InterruptedException e) {// 捕获拦截异常
102                 System.out.println("手术失败: " + e.getMessage());
103             }
104         }
105     }
106 }
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

【运行结果】程序运行结果如下。

```
病人 0 正在等待手术...
病人 1 正在等待手术...
病人 2 正在等待手术...
病人 0 开始执行手术...
手术开始,编号为 0
正在手术中~~
手术结束,编号为 0
病人 0 开始执行手术...
手术开始,编号为 1
正在手术中~~
病人 2 开始执行手术...
手术结束,编号为 1
手术开始,编号为 2
正在手术中~~
手术结束,编号为 2
```

【代码解析】 SurgeryThreadPool 类的 createTask ()方法根据任务编号执行指定的任务（第 02～09 行）。在类的 main()主方法中实例化三个 ThreadTask 对象，Thread 类的 sleep()方法使线程休眠 0.6 秒（第 03～16 行），运用循环执行创建的任务，调用 waitTaskColsed ()方法等待所有的任务执行完毕再关闭线程池（第 22、23 行）。

银行中的电话银行人工服务也可以作为线程池。例如，当我们打电话进行人工服务的时候，首先会提示进行等待，并尽快为你接通。而在银行的客服中心会有很多服务人员，在工作期间会有多个用户在同一时间拨打，例如，当一个服务人员正在通话的时候，其他的用户就只能等待，等通话结束之后才可以继续服务。此时的客服人员就相当于一个线程池，而拨打电话的用户就是线程。



17.6　Swing 线程应用

在图形界面开发的过程中，也会遇到不同的多线程问题。在本节中，就前面所涉及的线程问题运用到图形界面开发中。

17.6.1　多线程模拟钟表

使用多线程模拟一个可以走动的钟表，需要使用到多线程中的休眠方法。可以通过休眠方法来实现钟表秒针、分针和时针的时间刷新，从而完成从视觉上的钟表中时针、分针和秒针的走动。

【范例 17-13】示例代码 17-13 是一个钟表示例。

示例代码 17-13

```
01  public class MoveClock extends Applet implements Runnable {
02      Thread hhread = null;                    // 时针线程
03      Thread mThread = null;                   // 分针线程
04      Thread sThread = null;                   // 秒针线程
05  // 表示时针、分针、秒针端点的 XY 坐标点
06      int hourX, hourY, minuteX, minuteY, secondX, secondY;
07      int currentHour = 0;                     // 获取当前时间中代表小时的数字值
08      int currentMinute = 0;                   // 获取当前时间中代表分钟的数字值
09      int currentSecond = 0;                   // 获取当前时间中代表秒的数字值
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
10      Graphics sGraphics = null;          // 绘制秒针对象
11      Graphics mGraphics = null;          // 绘制分针对象
12      Graphics hGraphics = null;          // 绘制时针对象
13      Graphics2D mg2D = null;             // 创建 Graphics2D 分对象
14      Graphics2D hg2D = null;             // 创建 Graphics2D 小时对象
15      Graphics2D sg2D = null;             // 创建 Graphics2D 秒对象
16      double drawX[] = new double[61];    // 存放表盘刻度的 X 轴数组
17      double drawY[] = new double[61];    // 存放表盘刻度的 Y 轴数组
18      double dial_x[] = new double[61];   // 供绘制表盘使的 x 点坐标值
19      double dial_y[] = new double[61];   // 供绘制表盘使的 Y 点坐标值
20      int isRestart = 0;                  // 判断是否重新开始
21      public void init() {                // 数据初始化
22          hGraphics = this.getGraphics(); // 实例化时针 Graphics 对象
23          hGraphics.setColor(Color.white);// 设置时针的颜色
24          hg2D = (Graphics2D) hGraphics; // 实例化时针 Graphics2D 对象
25      hGraphics.translate(200, 200);      // 进行坐标系统变换原点设在(200,200)处
26      mGraphics = this.getGraphics();     // 实例化分针的 Graphics 对象
27      mg2D = (Graphics2D) mGraphics;      // 实例化分针的 Graphics2D 对象
28      mGraphics.setColor(Color.green);    // 设置分针的颜色
29      mGraphics.translate(200, 200); // 进行坐标系统变换，设置原点设在(200,200)处
30              sGraphics = this.getGraphics();         //实例化秒针 Graphics 对象
31      sg2D = (Graphics2D) sGraphics; // 实例化秒针 Graphics2D 对象
32      sGraphics.setColor(Color.blue);// 设置秒针的颜色
33      sGraphics.translate(200, 200); // 进行坐标系统变换，原点设在(200,200)处
34      drawX[0] = 0;
35          // 各个时针 12 点处的位置坐标（按新坐标系的坐标）
36          drawY[0] = -120;
37          dial_x[0] = 0;
38          // 12 点处的刻度位置坐标（按新坐标系的坐标）
39          dial_y[0] = -140;
40          double jiaodu = 6 * Math.PI / 180;
41      // 表盘分割成 60 份，将分割点的坐标存放在数组中
42      for (int i = 0; i < 60; i++) {
43          drawX[i + 1] = drawX[i] * Math.cos(jiaodu) - Math.sin(jiaodu)
44                  * drawY[i];
45          drawY[i + 1] = drawY[i] * Math.cos(jiaodu) + drawX[i]
46                  * Math.sin(jiaodu);
47      }
48      drawX[60] = 0;
49      drawY[60] = -120;
50      // 表盘分割成 60 份，将分割点的坐标存放在绘制数组中
51          for (int i = 0; i < 60; i++) {
52          dial_x[i + 1] = dial_x[i] * Math.cos(jiaodu) - Math.sin(jiaodu)
53                  * dial_y[i];
54          dial_y[i + 1] = dial_y[i] * Math.cos(jiaodu) + Math.sin(jiaodu)
55                  * dial_x[i];
56      }
57      dial_x[60] = 0;
58      dial_y[60] = -140;
59      }
60      public void start() {
61          if (isRestart >= 1) {
62              sThread.interrupt();        // 唤醒线程
63              mThread.interrupt();
64              hhread.interrupt();
65          }
66          hhread = new Thread(this);      // 创建时针线程
67          mThread = new Thread(this);     // 创建分针线程
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
68          sThread = new Thread(this);                    // 创建秒针线程
69          sThread.start();                               // 启动秒针线程
70          mThread.start();                               // 启动分针线程
71          hhread.start();                                // 启动时针线程
72          isRestart++;
73          if (isRestart >= 2)
74              isRestart = 1;
75      }
76      public void stop() {
77          sThread.interrupt();                           // 唤醒线程
78          mThread.interrupt();
79          hhread.interrupt();
80      }
81      public void paint(Graphics g) {                    // 绘制图形
82          this.setBackground(Color.black);
83          this.start();
84          g.drawOval(50, 50, 300, 300);                  // 表盘的外圈
85          g.translate(200, 200);                         // 进行坐标系统变换
86          for (int i = 0; i < 60; i++) {                 // 绘制表盘的小刻度和大刻度
87              if (i % 5 == 0) {
88                  g.setColor(Color.red);                 // 设置颜色
89                  g.fillOval((int) dial_x[i], (int) dial_y[i], 10, 10);
90              } else
91                  g.fillOval((int) dial_x[i], (int) dial_y[i], 5, 5);
92          }
93      }
94      public void run() {                                // 实现 Thread 的方法,开始线程
95          Date date = new Date();                        // 获取本地时间
96          String string = date.toString();
97          // 获得当前时间的小时
98          currentHour = Integer.parseInt(string.substring(11, 13));
99          // 获取当前时间的分钟
100         currentMinute = Integer.parseInt(string.substring(14, 16));
101         // 获取当前时间的秒钟
102         currentSecond = Integer.parseInt(string.substring(17, 19));
103         if (Thread.currentThread() == sThread) {       // 如果当前线程是秒线程
104             secondX = (int) drawX[currentSecond];      // 秒针初始化
105             secondY = (int) drawX[currentSecond];
106             // 用背景色清除前一秒的秒针
107             sGraphics.drawLine(0, 0, secondX, secondY);
108             sg2D.setStroke(new BasicStroke(2.0f)); // 设置所绘制秒针的宽度
109             int i = currentSecond;
110             while (true) {
111                 try {
112                     sThread.sleep(1000);               // 每隔一秒休眠
113                     Color c = getBackground();         // 获取背景颜色
114                     sGraphics.setColor(c);             // 设置秒针的颜色
115                     // 用背景色清除前一秒的秒针
116                     sGraphics.drawLine(0, 0, secondX, secondY);
117                     sg2D.setStroke(new BasicStroke(2.0f));
118                     // 秒针与分针重合,恢复分针显示
119                     if ((secondX == minuteX) && (secondY == minuteY)) {
120                         // 用背景色清除前一分的分针
121                         mGraphics.drawLine(0, 0, minuteX, minuteY);
122                     }
123                     // 秒针与时针重合，恢复时针的显示
124                     if ((secondX == hourX) && (secondY == hourY)) {
125                         // 用背景色清除前一时的时针
126                         hGraphics.drawLine(0, 0, hourX, hourY);
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
                    // 设置所绘制时针的宽度
                        hg2D.setStroke(new BasicStroke(4.0f));
                    }
                } catch (InterruptedException e) { // 捕获异常
                    Color c = getBackground();      // 获取背景颜色
                    sGraphics.setColor(c);          // 设置秒针的颜色
                    // 用背景色清除秒针
                    sGraphics.drawLine(0, 0, secondX, secondY);
                    sg2D.setStroke(new BasicStroke(2.0f));
                    return;
                }
                secondX = (int) drawX[(i + 1) % 60];    // 秒针向前走一个单位
            secondY = (int) drawY[(i + 1) % 60];    // 每一秒走 6°（一个单位格）
            sGraphics.setColor(Color.blue);         // 绘制秒针的颜色
                // 用背景色清除前一秒的秒针
                sGraphics.drawLine(0, 0, secondX, secondY);
                sg2D.setStroke(new BasicStroke(2.0f));
                i++;
            }
        }
        if (Thread.currentThread() == mThread) { // 如果当前线程是分线程
            minuteX = (int) drawX[currentMinute];
            minuteY = (int) drawY[currentMinute];
            mGraphics.drawLine(0, 0, minuteX, minuteY);
            mg2D.setStroke(new BasicStroke(3.0f));
            int i = currentMinute; // 获取当前分钟
            while (true) {
                try { // 第一次过 60-second 秒就前进一分钟，以后每过 60 秒前进一分钟
                    mThread.sleep(1000 * 60 - currentSecond * 1000);
                    currentSecond = 0;
                    Color c = getBackground();      // 获取背景颜色
                    mGraphics.setColor(c);          // 设置分针的颜色
            mg2D.setStroke(new BasicStroke(3.0f)); // 设置所绘制分针的宽度
            mGraphics.drawLine(0, 0, minuteX, minuteY);
            // 如果时针和分针重合
                if ((hourX == minuteX) && (hourY == minuteY)) {
                    hGraphics.drawLine(0, 0, minuteX, minuteY);
                    hg2D.setStroke(new BasicStroke(4.0f));
                }
            } catch (InterruptedException e) {
                return;
            }
            minuteX = (int) drawX[(i + 1) % 60];    // 分针向前走一个单位
            minuteY = (int) drawY[(i + 1) % 60];    // 每一分走 6°（一个单位格）
            mGraphics.setColor(Color.BLUE);         // 绘制分针的颜色
            mg2D.setStroke(new BasicStroke(3.0f));
            mGraphics.drawLine(0, 0, minuteX, minuteY);
            i++;
            currentSecond = 0;
        }
    }
    if (Thread.currentThread() == hhread) {          // 如果当前线程是时线程
        int h = currentHour % 12;
        hourX = (int) drawX[h * 5 + currentMinute / 12];
        hourY = (int) drawY[h * 5 + currentMinute / 12];
        int i = h * 5 + currentMinute / 12;
        hGraphics.drawLine(0, 0, hourX, hourY);
        hg2D.setStroke(new BasicStroke(4.0f));
        while (true) {
            try {
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
187      // 第一次过 12-minute%12 分钟就前进一个刻度，以后每过 12 分钟前进一刻度
188                  hhread.sleep(1000 * 60 * 12 - 1000 * 60
189                  * (currentMinute % 12) - currentSecond * 1000);
190                  currentMinute = 0;
191                  Color c = getBackground();
192                  hGraphics.setColor(c);
193                  hGraphics.drawLine(0, 0, hourX, hourY);
194                  hg2D.setStroke(new BasicStroke(4.0f));
195              } catch (InterruptedException e) {
196                  return;
197              }
198              hourX = (int) drawX[(i + 1) % 60];
199              hourY = (int) drawY[(i + 1) % 60];
200              hGraphics.setColor(Color.BLACK);
201              hGraphics.drawLine(0, 0, hourX, hourY);
202              hg2D.setStroke(new BasicStroke(4.0f));
203              i++;
204              currentMinute = 0;
205          }
206      }
207  }
208}
```

【运行结果】程序运行结果如图 17-1 所示。

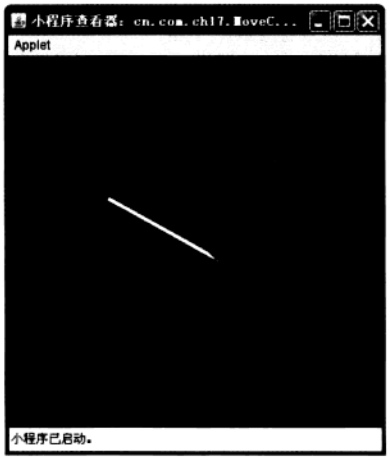


图 17-1　模拟钟表运行效果

17.6.2　线程的优先级模拟赛车

可以使用线程的优先级来模拟汽车比赛，优先级比较高的线程所获得的运行时间就会多一些，优先级低的线程所获得的运行时间相对来说就会少一点。在多线程开发中，并非每一个线程都会有相同的运行时间，所以，使用优先级就可以看出在一定的任务下，不同优先级所完成的进度不同。模拟一个赛车时，在相同的长度中，优先级不同，所使用的时间就会不同。

【范例 17-14】示例代码 17-14 是一个模拟赛车示例。

示例代码 17-14

```
01  public class Nascar extends JFrame {
02      private JPanel panel;
03      private BorderLayout borderLayout1 = new BorderLayout(); // 声明布局格式
04      private JLabel label = new JLabel();
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
    // 创建 JProgressBar 对象并实例化
    private JProgressBar bar_1 = new JProgressBar();
    private JProgressBar bar_2 = new JProgressBar();
    private JButton button_1 = new JButton();
    // 创建 Priority 类对象并实例化
    private Priority thread_1 = new Priority(bar_1, 10);
    private Priority thread_2 = new Priority(bar_2, 5);
    private void Initial() throws Exception {// 创建初始化程序，为其成员变量赋初值
        panel = (JPanel) this.getContentPane();
        label.setFont(new java.awt.Font("Dialog", 0, 14));
        label.setToolTipText("");
        label.setHorizontalAlignment(SwingConstants.CENTER);
        label.setText("赛车-线程的优先级(右边的优先级>左边的优先级)");
        panel.setLayout(borderLayout1);
        this.setSize(new Dimension(333, 232));
        this.setTitle("赛车比赛");
        bar_1.setOrientation(JProgressBar.VERTICAL);// 设置进度条的垂直方式
        // 设置进度条上显示字体的大小
        bar_1.setFont(new java.awt.Font("Dialog", 0, 14));
        // 设置该进度条的最大值
        bar_1.setMaximumSize(new Dimension(84, 32764));
        // 设置此组件的首选大小
        bar_1.setPreferredSize(new Dimension(126, 148));
        bar_1.setString("1 号参赛者");                    // 给进度条设置显示文字
        bar_1.setStringPainted(true);
        bar_2.setOrientation(JProgressBar.VERTICAL);
        bar_2.setFont(new java.awt.Font("Dialog", 0, 14));
        bar_2.setPreferredSize(new Dimension(126, 148));
        bar_2.setString("2 号参赛者");
        bar_2.setStringPainted(true);
        button_1.setFont(new java.awt.Font("Dialog", 0, 14));
        button_1.setToolTipText("");
        button_1.setText("开始");
        setSize(500, 500);
        setVisible(true);
        button_1.addActionListener(new java.awt.event.ActionListener() {
                //添加事件监听
            public void actionPerformed(ActionEvent e) {
                AddEvent(e);
            }
        });
        //将组件添加到面板上
        panel.add(label, BorderLayout.NORTH);
        panel.add(bar_1, BorderLayout.WEST);
        panel.add(bar_2, BorderLayout.EAST);
        panel.add(button_1, BorderLayout.SOUTH);
    }
    void AddEvent(ActionEvent e) {                    // 为按扭添加事件监听
        if (((JButton) e.getSource()).getText().equals("开始")) {
    // 当单击按钮文字是“Start”，则单击后设为“stop”，并且创建两个 Priority 的实例，
    //但这两个线程的优先级不同，然后才可以运行线程。
            this.button_1.setText("停止");
            Priority thread_1 = new Priority(this.bar_1, Thread.MAX_PRIORITY);
            thread_1.start();
            Priority thread_2 = new Priority(this.bar_2, Thread.MIN_PRIORITY);
            thread_2.start();
        } else {
            this.button_1.setText("开始");
            this.thread_1.stopped = true;
        }
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
65        }
66        public static void main(String[] args) {
67                // 该程序的入口处
68            try {
69                new Nascar().Initial();          // 调用该程序的 Initial 方法
70            } catch (Exception e) {
71                e.printStackTrace();
72            }
73        }
74    }
75    class Priority extends Thread {              // 设置线程优先级的类
76        JProgressBar bar_3;                      // 声明一个进度条对象
77        static boolean stopped;                  // 声明一个是否停止的标识
78        // 构造方法为成员变量赋初值
79        public Priority(JProgressBar bar_3, int priority) {
80            this.bar_3 = bar_3;
81            this.stopped = false;
82            this.setPriority(priority);
83        }
84        public void run() {                      // 启动线程
85            int min = 0;                         //设置最小值
86            int max = 1000;                      //设置最大值
87            this.bar_3.setMinimum(min);          //设置进度条的最小值
88            this.bar_3.setMaximum(max / 10);     //设置进度条的最大值
89            this.bar_3.setValue(min);            //设置进度条的当前值
90            for (int i = min; i <= max; i++) {
91                if (stopped) {
92                    break;
93                } else {
94                    this.bar_3.setValue((int) (i / 10));
95                    this.bar_3.setString(String.valueOf(i));
96                    try {
97                        Thread.sleep(2);
98                    } catch (InterruptedException e) {
99                        e.printStackTrace();
100                   }
101               }
102           }
103       }
104   }
```

【运行结果】运行结果如图 17-2 所示。

17.6.3 使用断点实现计数器

使用多线程的休眠和中断运行线程的方法可以实现一个计数器，在这个计数器中，主要由两个按钮和一个标签组成，两个按钮分别是“开始”和“停止”，单击“开始”按钮就可以使标签中的数字开始增加计时，单击“停止”按钮就会调用线程的中断，并且此时数字停止。再次单击“开始”，则会使时间从零开始重新计时。

【范例 17-15】示例代码 17-15 是一个计数器示例。

示例代码 17-15

```
01  public class DigitalTimer extends JLabel implements ActionListener {
02      private volatile String value;      // 要在文本框中显示的内容
03      private Thread thread;              // 声明一个线程
04      private volatile boolean isStop;    // 标识是否停止计时
05      Runnable r = null;                  //声明一个 Runnable 子类的对象
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

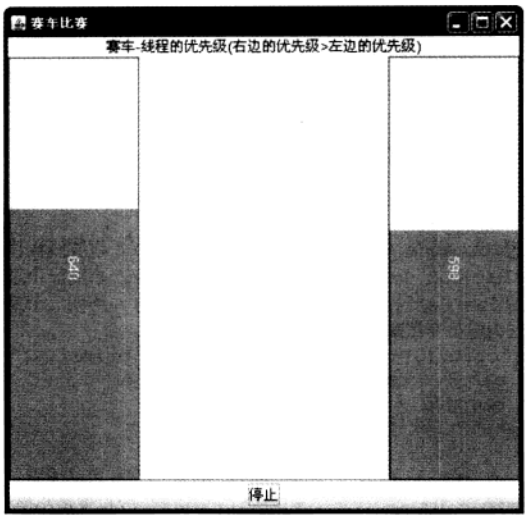


图 17-2　赛车运行效果

```
06      public DigitalTimer() {                  // 构造方法为 Swing 组件初始化
07          setBorder(BorderFactory.createLineBorder(Color.blue));
08          setHorizontalAlignment(SwingConstants.RIGHT);
09          this.setForeground(Color.red);
10          setFont(new Font("SansSerif", Font.BOLD, 16));
11          setText("00000.0");                  //设置 component 的大小
12          setMinimumSize(getPreferredSize());
13          setPreferredSize(getPreferredSize());
14          setSize(getPreferredSize());
15          value = "0.0";
16          setText(value);
17          isStop = true;
18          r = new Runnable() {                 // 创建一个实现 Runnable 接口内部类对象
19              public void run() {
20                  try {
21                      startTime();
22                  } catch (Exception x) {
23                      x.printStackTrace();
24                  }
25              }
26          };
27          thread = new Thread(r, "Stopwatch");// 根据 Runnable 对象来创建一个线程
28          thread.start();                      // 启动线程
29      }
30      private void startTime() {               // 开始计时
31          long startTime = System.currentTimeMillis();// 线程开始运行的时间
32          int n = 0;                           // 一个时间量
33          long sleepTime = 100;                // 线程正在休眠的时间
34          long nextSleepTime = 100;            // 接下来准备休眠的时间
35          DecimalFormat format = new DecimalFormat("0.0");
36          Runnable updateText = new Runnable() {
37              public void run() {
38                  setText(value);
39              }
40          };
41          while (isStop) {
42              try {
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
43                  Thread.sleep(nextSleepTime);
44                  n++;
45                  // 获取系统的当前时间
46                  long nowTime = System.currentTimeMillis();
47                  // 经过的时间=当前时间-开始时间
48                  long elapsedTime = nowTime - startTime;
49                  nextSleepTime = sleepTime + ((n * 100) - elapsedTime);
50                  if (nextSleepTime < 0) {
51                      nextSleepTime = 0;
52                  }
53                  value = format.format(elapsedTime / 1000.0);
54                  System.out.println(value);
55                  // 使该线程在 Swing 组件上同步执行
56                  SwingUtilities.invokeAndWait(updateText);
57              } catch (InterruptedException ix) {
58                  return;                              // 停止运行
59              } catch (InvocationTargetException x) {
60                  x.printStackTrace();
61              }
62          }
63      }
64      public void endTime() {                          // 结束计时
65          isStop = false;
66          thread.interrupt();                          // 中断线程
67      }
68      public boolean isAlive() {
69          return thread.isAlive();                     // 判断该线程是否是活动的
70      }
71      public static void main(String[] args) {
72          DigitalTimer dt = new DigitalTimer();        // 创建一个本类的实例化对象
73          // 创建一个面板对象，其布局方式为流布局
74          JPanel p = new JPanel(new FlowLayout());
75          p.add(dt);                                   // 将类对象添加到面板中
76          JFrame frame = new JFrame("数字计时器");// 创建一个框架
77          // 创建 2 个 JButton 按钮，其作用是开始计时和停止计时，并为其添加相应的事件处理
78          JButton b1 = new JButton();
79          b1.setText("开始");                          // 为接钮设置文本文字
80          b1.setBounds(20, 40, 60, 20);                // 设置按钮的大小
81          b1.setActionCommand("start");                // 设置激发此按钮的事件命令名称
82          b1.addActionListener(dt);                    // 添加监听事件
83          JButton b2 = new JButton();
84          b2.setText("停止");
85          b2.setBounds(80, 40, 60, 20);
86          b2.setActionCommand("end");                  // 设置激发此按钮的事件命令名称
87          b2.addActionListener(dt);                    // 添加监听事件
88          frame.setContentPane(p);                     // 在该框架中添加面板
89          frame.add(b1);                               // 在该框架中添加按钮组件
90          frame.add(b2);
91          frame.setSize(250, 100);                     // 设置框架的大小
92          frame.setVisible(true);                      // 设置框架为可见
93      }
94      public void actionPerformed(ActionEvent e) {// 事件处理
95          if (e.getActionCommand().equals("start")) { // 如果激发开始按钮
96              if (!isAlive()) {                        // 判断如果当前的线程不处于可活动状态
97                  isStop = true;                       // 将状态设置为 true
98                  //重新根据 Runnable 对象来创建一个新的线程
99                  thread = new Thread(r, "Stopwatch");
100                 thread.start();                      // 启动新线程
101             } else {
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
102                     startTime();
103                 }
104             } else if (e.getActionCommand().equals("end")) {// 如果激发停止按钮
105                 endTime();    // 则调用 endTime 方法，中断当前正在运行的线程
106             }
107         }
108 }
```

【运行结果】程序运行结果如图 17-3。

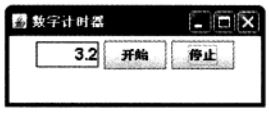


图 17-3　数字计数器

【代码解析】通过内部类创建了一个实现 Runnable 接口的子类，并重写 run 方法，在其 run 方法中调用 startTime()方法（第 18～26 行）。根据 Runnable 对象来创建一个线程 thread，并调用 start 方法启动线程。startTime()方法的主要作用是进行计时操作，首先通过 System.currentTimeMillis()获得线程首次开始运行的时间，然后创建一个实现 Runnable 接口的对象 updateText，在其 run 方法中调用 setText()方法（第 30～63 行）。

17.6.4　使用线程实现屏保“璀璨星空”

在 Windows XP 操作系统中，有一个屏幕保护程序叫“璀璨星空”，这个效果可以使用线程来实现。

【范例 17-16】示例代码 17-16 是一个璀璨星空示例。

示例代码 17-16

```
01  public class Universe extends java.applet.Applet implements Runnable {
02      int Width, Height;              // 定义小程序的长和宽
03      Thread thread = null;           // 声明一个线程对象
04      boolean suspend = false;        // 是否暂停
05      Image im;                       // 声明一个图像对象
06      Graphics graphics;              // 声明一个 Graphics 对象
07      double rot, dx, ddx;            // 声明 double 型变量
08      int speed, stars, type;         // 声明 int 型变量
09      double defddx, max;             // 声明 double 型变量
10      Star pol[];                     // 星光
11      public void init() {            // 初始化 Applet 小程序
12          rot = 0;
13          dx = 0;
14          ddx = 0;
15          Width = 300;
16          Height = 300;
17          String theSpeed = "25";
18          Show("speed", theSpeed);
19      speed = (theSpeed == null) ? 50 : Integer.valueOf(theSpeed).intValue();
20      String theStars = "250";
21      Show("stars", theStars);
22      stars = (theStars == null) ? 30 : Integer.valueOf(theStars).intValue();
23          try {
24              im = createImage(Width, Height);
25              graphics = im.getGraphics();
26          } catch (Exception e) {
27              graphics = null;
28          }
29          pol = new Star[stars];
30          for (int i = 0; i < stars; i++)
31              pol[i] = new Star(Width, Height, 150, type);
32          System.out.println(Width + "," + Height);
33      }
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
34      public void paint(Graphics g) {     // 绘制组件
35          if (graphics != null) {
36              paintStart(graphics);
37              g.drawImage(im, 0, 0, this);
38          } else {
39              paintStart(g);
40          }
41      }
42      public void paintStart(Graphics g) {
43          g.setColor(Color.black);
44          g.fillRect(0, 0, Width, Height);
45          for (int i = 0; i < stars; i++)
46              pol[i].DrawSelf(g, rot);
47      }
48      public void start() {                          // 启动 Applet 小程序
49          if (thread == null) {
50              thread = new Thread(this);
51              thread.start();                        // 启动线程
52          }
53      }
54      public void stop() {                           // 停止运行 Applet 小程序
55          if (thread != null) {
56              thread.stop();
57              thread = null;
58          }
59      }
60      public void run() {                            // 运行线程
61          while (thread != null) {
62              rot += dx;
63              dx += ddx;
64              if (dx > max)
65                  ddx = -defddx;
66              if (dx < -max)
67                  ddx = defddx;
68              try {
69                  Thread.sleep(speed);
70              } catch (InterruptedException e) {
71              }
72              repaint();
73          }
74      }
75      public void update(Graphics g) {               // 重新绘制组件
76          paint(g);
77      }
78      public void Show(String theString, String theValue) {
79          if (theValue == null) {
80              System.out.println(theString + " : null");
81          } else {
82              System.out.println(theString + " : " + theValue);
83          }
84      }
85  }
86  class Star {                                       // 代表星星类
87      int H, V;
88      int x, y, z;
89      int type;
90      Star(int width, int height, int depth, int type) {// 构造函数为各变量初始化
91          this.type = type;
92          H = width / 2;
93          V = height / 2;
94          x = (int) (Math.random() * width) - H;
95          y = (int) (Math.random() * height) - V;
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
96              if ((x == 0) && (y == 0))
97                  x = 10;
98              z = (int) (Math.random() * depth);
99          }
100         public void DrawSelf(Graphics g, double rot) {  // 根据坐标绘制星星
101             double X, Y;
102             int h, v, hh, vv;
103             int d;
104             z -= 2;
105             if (z < -63)
106                 z = 100;
107             hh = (x * 64) / (64 + z);
108             vv = (y * 64) / (64 + z);
109             X = (hh * Math.cos(rot)) - (vv * Math.sin(rot));
110             Y = (hh * Math.sin(rot)) + (vv * Math.cos(rot));
111             h = (int) X + H;
112             v = (int) Y + V;
113             if ((h < 0) || (h > (2 * H)))
114                 z = 100;
115             if ((v < 0) || (v > (2 * H)))
116                 z = 100;
117             setColor(g);
118             if (type == 0) {
119                 d = (100 - z) / 50;
120                 if (d == 0)
121                     d = 1;
122                 g.fillRect(h, v, d, d);
123             } else {
124                 d = (100 - z) / 20;
125                 g.drawLine(h - d, v, h + d, v);
126                 g.drawLine(h, v - d, h, v + d);
127                 if (z < 50) {
128                     d /= 2;
129                     g.drawLine(h - d, v - d, h + d, v + d);
130                     g.drawLine(h + d, v - d, h - d, v + d);
131                 }
132             }
133         }
134         public void setColor(Graphics g) {                  // 给绘制的对象设置颜色
135             if (z > 50) {
136                 g.setColor(Color.gray);
137             } else if (z > 25) {
138                 g.setColor(Color.lightGray);
139             } else {
140                 g.setColor(Color.white);
141             }
142         }
143 }
```

【运行结果】程序运行结果如图 17-4 所示。



图 17-4　璀璨星空效果

17.6.5　实现下雪场景

本示例是一个实现空中飘雪的场景，它运用了多线程休眠方法。该示例和前面的璀璨星空效果有所不同，在其中的雪花是向下飘落的，而前面的璀璨星空则是向外扩散。

【范例 17-17】示例代码 17-17 是一个下雪示例。

示例代码 17-17

```
01  public class SnowVillage extends Applet implements Runnable {
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
    Thread thread;                          // 声明主线程
    Image off, images[];                    // 声明图片对象和图片数组
    Random random;                          // 声明一个随机数对象
    int flag, sonwNum, wind, thread_1, size;    // 声明一些 int 型变量
    int[] X, Y;                             // 声明两个 int 型数组，分别表示 X 和 Y 坐标点
    long time = 0;                          // 表示时间
    Dimension ds;                           // 声明一个 Dimension 对象
    MediaTracker mt   ;                     // 声明一个 MediaTracker 对象
    // 获取 HTML 标记中命名参数的值
    int getValue(String s1, int s2, int max, int min) {
        String s = getParameter(s1);
        if (s != null) {
            if ((s2 = Integer.parseInt(s)) > max)
                return max;
            else if (s2 < min)
                return min;
            else
                return s2;
        } else
            return s2;
    }
    public void init() {                                    // Applet 小程序初始化
        this.setSize(300, 200);
        random = new Random();
        ds = getSize();
        off = createImage(ds.width, ds.height);     // 创建一个图像
        sonwNum = getValue("sonwNum", 100, 500, 0); // 获取雪片的个数
        size = getValue("size", 3, 10, 3);          // 获取雪片的大小
        thread_1 = getValue("threadsleep", 80, 1000, 10);// 获取休眠的时间
        // 获取绘制雪片的 XY 坐标值
        X = new int[sonwNum];
        Y = new int[sonwNum];
        for (int i = 0; i < sonwNum; i++) {
            X[i] = random.nextInt() % (ds.width / 2) + ds.width / 2;
            Y[i] = random.nextInt() % (ds.height / 2) + ds.height / 2;
        }
        mt = new MediaTracker(this);
        images = new Image[1];
        images[0] = getImage(getDocumentBase(), "xue.jpg");
        mt.addImage(images[0], 0);
        try {
            mt.waitForID(0);
        } catch (InterruptedException _ex) {
            return;
        }
        flag = 0;
    }
    public void start() {                                   // 启动小程序
        if (thread == null) {
            thread = new Thread(this);
            thread.start();                                 // 启动线程
        }
    }
    public void stop() {                                    // 停止运行小程序
        thread = null;
    }
    public void run() {                                     // 运行线程
        while (thread != null) {
            try {
                Thread.sleep(thread_1);
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
62                } catch (InterruptedException _ex) {
63                    return;
64                }
65                repaint();
66            }
67        }
68        public void snow(Graphics g) {                    // 绘制雪片
69            g.setColor(Color.white);
70            for (int i = 0; i < sonwNum; i++) {
71                g.fillOval(X[i], Y[i], size, size);
72                X[i] += random.nextInt() % 2 + wind;
73                Y[i] += (random.nextInt() % 6 + 5) / 5 + 1;
74                if (X[i] >= ds.width)
75                    X[i] = 0;
76                if (X[i] < 0)
77                    X[i] = ds.width - 1;
78                if (Y[i] >= ds.height || Y[i] < 0) {
79                    X[i] = Math.abs(random.nextInt() % ds.width);
80                    Y[i] = 0;
81                }
82            }
83            wind = random.nextInt() % 5 - 2;
84        }
85        public void paint(Graphics g) {                   // 绘制组件
86            off.getGraphics().setColor(Color.black);
87            off.getGraphics().fillRect(0, 0, ds.width, ds.height);
88            off.getGraphics().drawImage(images[0], 0, 0, this);
89            snow(off.getGraphics());
90            g.drawImage(off, 0, 0, null);
91        }
92        public void update(Graphics g) {                  // 重新绘制组件
93            paint(g);
94        }
95    }
```

【运行结果】程序运行结果如图 17-5 所示。



图 17-5　下雪场景效果

【代码解析】实现 Runnable 接口的 run 方法在当线程每休眠一段时间后调用 repaint()重新绘制雪花（第 58～66 行）。snow()方法的主要作用是绘制雪花，在该方法中首先通过 setColor 设置雪花的颜色为白色，并利用随机数来确定绘制下雪的位置（第 68～84 行）。

17.6.6　实现瀑布效果

相信大家在电视或者一些图画上都见到过高山瀑布。在学习了多线程之后，我们可以使用多线程来模拟一个瀑布的效果。本示例就是一个实现瀑布流水的效果。

【范例 17-18】示例代码 17-18 是一个模拟瀑布效果示例。

示例代码 17-18

```
01  public class Faucet extends Applet implements Runnable {
02      final int Max = 1000;       // 水滴的最大个数
03      Drop d[];                   // 声明一个水滴对象
04      int width, height, X, Y;    // 声明 Applet 小程序的宽和高，以 XY 坐标中心
05      Image off;                  // 声明一个图片对象
06      Graphics graphics;          // 声明一个 Graphics 对象
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
07      Thread thread;              // 声明一个线程对象
08      public void init() {        // Applet 小程序初始化
09          this.setSize(300, 200);
10          setBackground(Color.gray);
11          width = getSize().width;
12          height = getSize().height;
13          d = new Drop[Max];
14          for (int i = 0; i < Max; i++)
15              d[i] = new Drop();
16          off = createImage(width, height);
17          graphics = off.getGraphics();
18      }
19      public void start() {       // 开始执行小程序
20          thread = new Thread(this);
21          thread.start();
22      }
23      public void stop() {                                  // 停止执行小程序
24          thread = null;
25      }
26      public void update(Graphics g) {                      // 重新绘制小程序
27          paint(g);
28      }
29      public void paint(Graphics g) {                       // 绘制组件
30          g.drawImage(off, 0, 0, this);
31      }
32      public void run() {                                   / 运行线程
33          boolean reset = false;
34          int i, t = 0;
35          while (true) {
36              graphics.clearRect(0, 0, width, height);//清除指定矩形的背影色
37              graphics.setColor(Color.white);          //设置当前的颜色为白色
38              graphics.drawLine(0, 15, 10, 15);        //画一条直线
39              for (i = 0; i < Max; i++) {
40                  graphics.fillOval((int) d[i].X, (int) d[i].Y, 3, 3);
41                  d[i].X = d[i].X + d[i].newX;
42                  if (d[i].X > 10) {
43                      d[i].Y += d[i].newY * d[i].time / 1000;
44                      d[i].newY = (int) 9.8 * d[i].time;
45                      d[i].time++;
46                  }
47                  if (d[i].Y > height) {
48                      d[i].reset();
49                  }
50              }
51              repaint();                              //重新绘制组件
52              try {
53                  Thread.sleep(100);                  //线程休眠 100ms
54              } catch (InterruptedException e) {
55              }
56          }
57      }
58  }
59  class Drop {                                          // 水滴类
60      double X, Y;                                      //声明坐标 x 和 y
61      double newX, newY;                                //声明新坐标
62      int time;
63      public Drop() {
64          reset();
65      }
66      public void reset() {                             // 重新设置绘制水滴的位置和大小
67          X = (int) (Math.random() * -40);
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
68            Y = (int) (Math.random() * 5 + 10);
69            newX = Math.random() * 3 + 1.0;
70            newY = 0;
71            time = 0;
72        }
73    }
```



图 17-6　瀑布效果图

【运行结果】程序运行结果如图 17-6 所示。

17.6.7　滚动字幕

在平时的广告牌上经常可以看到 LED 的显示屏显示有滚动的文字广告，或者在一些出租车后面也有这样的显示屏。在这里可以使用前面所学习的多线程知识来模拟一个实现文字滚动的滚动字幕。

【范例 17-19】示例代码 17-19 是一个多线程死锁示例。

示例代码 17-19

```
01  public class CryptoService extends JComponent {
02      private BufferedImage image;                    // 声明一个 BufferedImage 类对象
03      private Dimension imageOfSize;                  // 声明一个 Dimension 类对象
04      private volatile int currOffset;                // 声明一个用于表示偏移量的变量
05      private Thread thread;                          // 声明一个线程
06      private volatile boolean flag;
07      public CryptoService(String text) {             //构造方法为各成员变量赋值
08          currOffset = 0;                             //设置当前的值
09          buildImage(text);                           //调用自定义方法 buildImage()
10          setMinimumSize(imageOfSize);                //设置组件的最小尺寸
11          setPreferredSize(imageOfSize);              //设置组件的首选尺寸
12          setMaximumSize(imageOfSize);                //设置组件的最大尺寸
13          setSize(imageOfSize);
14          flag = true;
15          Runnable r = new Runnable() {               //实现接口内部类
16              public void run() {
17                  try {
18                      ScrollING();
19                  } catch (Exception x) {
20                      x.printStackTrace();
21                  }
22              }
23          };
24          thread = new Thread(r, "ScrollText");
25          thread.start();
26      }
27      private void buildImage(String text) {
28          //根据抗锯齿提示键和抗锯齿提示值来创建一个 RenderingHints 类对象
29          RenderingHints renderHints = new RenderingHints(
30                  RenderingHints.KEY_ANTIALIASING,
31                  RenderingHints.VALUE_ANTIALIAS_ON) ;
32          //将指定的呈现提示键映射到此 RenderingHints 对象中的指定呈现指示值。
33          renderHints.put(RenderingHints.KEY_RENDERING,
34                  RenderingHints.VALUE_RENDER_QUALITY);
35          //构造一个类型为预定义的图像
36          BufferedImage scratchImage = new BufferedImage(1, 1,
37                  BufferedImage.TYPE_INT_RGB);
38          //创建一个 Graphics2D 类对象
39          Graphics2D scratchG2 = scratchImage.createGraphics();
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
40          //设置呈现算法
41          scratchG2.setRenderingHints(renderHints);
42          //创建一个 Font 字体对象
43          Font font = new Font("Serif", Font.BOLD | Font.ITALIC, 24);
44          //创建一个 FontRenderContext 对象
45          FontRenderContext frc = scratchG2.getFontRenderContext();
46          //创建一个 TextLayout 对象
47          TextLayout tl = new TextLayout(text, font, frc);
48          //创建一个 Rectangle2D 对象
49          Rectangle2D textBounds = tl.getBounds();
50          // 设置文字的显示宽度
51          int textWidth = (int) Math.ceil(textBounds.getWidth());
52          // 设置文字的显示长度
53          int textHeight = (int) Math.ceil(textBounds.getHeight());
54          int horizontalPad = 10;                  // 设置水平间距为 10 像素
55          int verticalPad = 6;                     // 设置垂直间距为 6 像素
56          imageOfSize = new Dimension(             // 创建 Dimension 对象
57                  textWidth + horizontalPad, textHeight + verticalPad);
58          image = new BufferedImage(
59                  imageOfSize.width, imageOfSize.height,
60                  BufferedImage.TYPE_INT_RGB);     // 创建 BufferedImage 对象
61          Graphics2D g2 = image.createGraphics();
62          g2.setRenderingHints(renderHints);
63          int baselineOffset = (verticalPad / 2) - ((int) textBounds.getY());
64              // 基线的偏移量
65          g2.setColor(Color.black);                // 设为黑色背景
66          g2.fillRect(0, 0, imageOfSize.width, imageOfSize.height);
67          g2.setColor(Color.WHITE);                // 白色字体
68          tl.draw(g2, 0, baselineOffset);
69          scratchG2.dispose();
70          scratchImage.flush();
71          g2.dispose();
72      }
73      public void paint(Graphics g) {
74          // 将当前的剪贴区设置为由给定坐标指定的矩形。
75          g.setClip(0, 0, imageOfSize.width, imageOfSize.height);
76          int localOffset = currOffset;            // 本地偏移量是不断变化的
77          g.drawImage(image, -localOffset, 0, this);
78          g.drawImage(image, imageOfSize.width - localOffset, 0, this);
79          // 绘制出边框
80          g.setColor(Color.red);
81          g.drawRect(0, 0, imageOfSize.width - 1, imageOfSize.height - 1);
82      }
83      private void ScrollING() {                   // 执行滚动操作
84          while (flag) {
85              try {
86                  Thread.sleep(100);
87                  currOffset = (currOffset + 1) % imageOfSize.width;
88                  repaint();
89              } catch (InterruptedException x) {
90                  Thread.currentThread().interrupt();
91              }
92          }
93      }
94      public static void main(String[] args) {
95          //创建 CryptoService 类对象
96          CryptoService st = new CryptoService("大家好....");
97          JPanel p = new JPanel(new FlowLayout());     //定义一个有布局的面板对象
98          p.add(st);
99          JFrame f = new JFrame("滚动的文字");          //创建一个框架对象
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
100             f.setContentPane(p);
101             f.setSize(400, 100);
102             f.setVisible(true);
103         }
104 }
```

【运行结果】运行结果如图 17-7 所示。

【代码解析】以上程序中，通过内部类创建了一个实现 Runnable 接口的线程对象，该线程的主要作用是调用 ScrollING()方法（第 15～23 行）。在 buildImage()方法中首先根据抗锯齿提示键和抗锯齿提示值创建一个 RenderingHints 类对象，并将指定的呈现提示键映射到此 RenderingHints 对象中的指定呈现指示值（第 29～34 行）。根据要进行滚动的文字和设置好的字体来创建一个 TextLayout 对象，TextLayout 可以移动和定位光标，也可以根据样式排列对文本进行分段。在该方法中，将滚动文字的背景色设置为黑色，文字的颜色设置为白色（第 47～67 行）。在 ScrollING()方法中，通过 Thread 类的 sleep()方法使文字每隔 0.1 秒便向右移动一个像素（第 83～93 行）。

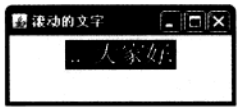


图 17-7　滚动的文字

17.7　习题

【习题 17-1】编写一个多线程程序，定义两个不同优先级的线程，通过使用调度方法，使优先级较低的线程获得更多的运行机会。

【分析】该题主要考查多线程的优先级使用。

【核心代码】

```
class MyRunnable1 implements Runnable                    //实现 Runnable 接口
{
    public void run() {                                  //线程执行的时候打印出加号
        for(int i=0;i<200;i++)                           //每个线程打印 200 个加号
            System.out.print("+");
    }
}
class MyRunnable2 implements Runnable                    //实现 Runnable 接口
{
    public void run() {                                  //线程执行的时候打印出星号
        for(int i=0;i<200;i++)                           //每个线程打印 200 个星号
            System.out.print("*");
    }
}
public class Demo2 {
    public static void main(String[] args) {
        //创建两个实现了 Runnable 接口的类的对象
        MyRunnable1 my1=new MyRunnable1();
        MyRunnable2 my2=new MyRunnable2();
        //创建两个线程
        Thread th1=new Thread(my1);
        Thread th2=new Thread(my2);
        //取得两个线程的优先级
        System.out.print("线程 th1 的优先级为：");
        System.out.println(th1.getPriority());
        System.out.print("线程 th2 的优先级为：");
        System.out.println(th2.getPriority());
        //给 th1 最高的优先级
        th1.setPriority(Thread.MAX_PRIORITY);
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
        //给 th2 最低的优先级
        th2.setPriority(Thread.MIN_PRIORITY);
        System.out.print("改变优先级后线程 th1 的优先级为: ");
        System.out.println(th1.getPriority());
        System.out.print("改变优先级后线程 th2 的优先级为: ");
        System.out.println(th2.getPriority());
        //启动线程
        th1.start();
        th2.start();
    }
}
```

【习题 17-2】编写一个多线程程序，模拟一个参赛者在参加比赛的过程中，以休息、参加比赛和处理一些突发事件等情况来介绍如何使线程处于休眠状态，又如何将休眠的线程重新唤醒。

【分析】本题主要结合多线程和 Swing 图形界面开发模拟一个比赛过程，类似于进度条，使用将线程休眠以及唤醒方法对 Swing 进行操作。

【核心代码】

```
private DateFormat df = new SimpleDateFormat("ss:SS");
    public static void main(String[] args) {    // java 程序主入口处
        Relay relay = new Relay();               // 实例化对象
        relay.start();                           // 启动线程
        try {
            relay.join();                        // 等待线程运行结束
        } catch (InterruptedException e) {       // 捕获唤醒异常
            System.out.println(" 收到命令，准备上场: " + e.getMessage());
        }
        relay.incident();                        // 调用方法判断是否唤醒
    }
    public void incident() {
        Thread.currentThread().interrupt();      // 唤醒当前线程
        while (true) {
            if (Thread.currentThread().isInterrupted()) {// 判断当前线程是否被唤醒
            System.out.print(df.format(new Date()) + "  现在是否正在准备上场? ");
            System.out.println(Thread.currentThread().isInterrupted() ? "是"
                        : "没有");
                try {
                    Thread.currentThread().sleep(3000); // 线程休眠 3s
                } catch (InterruptedException e) {       // 捕获唤醒异常
                System.out.println(df.format(new Date()) + " 收到命令，停止休息: "
                            + e.getMessage());
                }
                System.out.print(df.format(new Date()) + "  比赛结束后是否参加下一
                轮比赛? ");
                System.out.println(Thread.currentThread().isInterrupted()?"是"
                        : "不参加");
            }
        }
    }
    public void run() {
        System.out.println("第一场比赛结束的时间为: " + df.format(new Date()));
        System.out.println("休息 5 小时");
        try {
            sleep(2000);                         // 线程休眠 2s，程序假设 1s=1h
        } catch (InterruptedException e) { // 捕获唤醒异常
            System.out.println(df.format(new Date()) + "收到命令，准备上场: "
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
                    + e.getMessage());
            }
            System.out.print(df.format(new Date()) + "在休息的过程中是否又参加其他的比赛?
            ");
            try {
                sleep(2000);                    // 线程休眠 2s
            } catch (InterruptedException e) { // 捕获唤醒异常
                System.out.println(df.format(new Date()) + "收到命令，准备上场："
                        + e.getMessage());
            }
            System.out.println(!isAlive() ? "参加比赛" : "没有参加其他的比赛");
                        // 线程是否激活,false 表示不是激活的
            interrupt();// 唤醒线程
            System.out.print(df.format(new Date()) + " 休息中，替补队员受伤，是否参加比赛?
            ");
            System.out.println(isAlive() ? "参加比赛" : "不参加比赛"); // 线程是否激活
}
```

【习题 17-3】编写一个程序设计一个定时器，比如：在开发一个企业考勤管理系统时，需要每天在 9:00 之前和 17:30 之后这两个时间段来查看员工的打卡时间，以此判断是否有人迟到和早退。

【分析】该题主要考查对线程各个知识点的综合使用。

【核心代码】

```
public static void main(String[] args) {    // java 程序主入口处
        Timer timer = new Timer();           // 创建定时器类
        TimerTask tt1 = new MyTask(1);
        timer.schedule(tt1, 200);            // 0.2s 后执行任务
        TimerTask tt2 = new MyTask(2);
        timer.schedule(tt2, 500, 1000);      // 0.5s 后执行任务，并每隔 1s 执行一次
        TimerTask tt3 = new MyTask(3);
        Date date = new Date(System.currentTimeMillis() + 1000);
        timer.schedule(tt3,.date);           // 在指定时间 1s 后执行任务
        try {
            Thread.sleep(3000);              // 休眠 3s 钟
        } catch (InterruptedException e) { // 捕获拦截异常
            System.out.println("出现错误：" + e.getMessage());
        }
        timer.cancel();                      // 终止定时器，取消定时器中的任务
        System.out.println("任务定时器已经被取消");
    }
}
class MyTask extends TimerTask {             // 继承时间任务类执行任务
    private int taskID = 0;                  // 任务编号
    public MyTask(int id) {                  // 带参数的构造方法进行初始化
        this.taskID = id;
    }
    public void run() {                      // 实现 TimerTask 中的方法
        System.out.println("开始执行我的第" + this.taskID + "个任务 ，执行时间为"
                + new Date().toLocaleString());
    }
```

【习题 17-4】创建一个 Guard 类，此类每隔 5 秒会查看一下安全情况。

【分析】该题主要考查线程的创建和睡眠。

【核心代码】

```
public class Guard implements Runnable {
   public void run() {
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
        for (int i = 0; i < 20; i++) {
            System.out.println("检查安全");
            try {
                sleep(5000);
            } catch (InterruptedException e) {
                System.out.println();
            }
        }
    }

    public static void main(String[] args) {
        Thread guard = new Thread(new Guard());
        guard.start();

    }
}
```

【习题 17-5】编写一个程序模拟大学生上自习，到中午的时候去吃饭，然后回来继续上自习。

【分析】该题主要考查线程的创建与合并。

【核心代码】

```
public class Lunch implements Runnable{
    public void run() {
        for (int i = 0; i < 3; i++) {
            System.out.println("正在吃午饭");
            try {
                Thread.sleep(3000);
            } catch (InterruptedException e) {
                System.out.println();
            }
        }
    }
}

public class Student implements Runnable {
    public Thread lunch = new Thread(new Lunch());
    public void run() {
        for (int i = 0; i < 20; i++) {
            System.out.println("正在上自习" + i);
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                System.out.println();
            }
            if (i == 10) {
                lunch.start();
                try {
                    lunch.join();
                } catch (InterruptedException e) {
                    System.out.println("---");
                }
            }
        }
    }

    public static void main(String[] args) {
        Thread student = new Thread(new Student());
        student.start();
    }
}
```

【习题 17-6】编写一个程序模拟汽车在公路上行驶，当这辆车没有牌照时，警察就会把这

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

辆车拦下。

【分析】该题主要考查线程的创建与让步。

【核心代码】

```
public class Car implements Runnable{
    private boolean stoped = false;
    private String license;
    public Car() {}

    public Car(String lic) {
        this.license = lic;
    }

    public void run() {
        for (int i = 0; i < 20 && !stoped; i++) {
            System.out.println((license == null ? "无牌照汽车" : license) + "正在行驶
            中...");
            if (i == 11) {
                System.out.println((license == null ? "无牌照汽车" : license) + "被警
                察检查");
                if (null == license) {
                    System.out.println("无牌照汽车被拦下");
                    stoped = true;
                } else {
                    Thread.yield();
                }
            }
        }
    }

    public static void main(String[] args) {
        Thread c1 = new Thread(new Car("123"));
        Thread c2 = new Thread(new Car());
        Thread c3 = new Thread(new Car("789"));
        c1.start();
        c2.start();
        c3.start();
    }
}
```

【习题 17-7】编写一个程序模拟学生在食堂买饭，一个人的素菜、荤菜、米饭或馒头买完后，下一个才能买。

【分析】该题主要考查多线程的同步。

【核心代码】

```
public class Canteen{
    synchronized public void sell(String student, String... foods) {
        for (String f : foods) {
            System.out.println(student + "买" + f);
        }
    }
}

public class Worker implements Runnable {
    private String name;
    private Canteen canteen;
    private String[] foods;

    public Worker(String name, Canteen canteen, String... foods) {
        this.name = name;
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
        this.canteen = canteen;
        this.foods = foods;
    }
    public void run() {
        canteen.sell(name, foods);
    }

    public static void main(String[] args) {
        Canteen c = new Canteen();
        Thread w1 = new Thread(new Worker("Jack", c, "potato", "meat", "rice"));
        Thread w2 = new Thread(new Worker("Mike", c, "eggplant", "egg", "bread"));
        Thread w3 = new Thread(new Worker("Rose", c, "cabbage", "egg", "rice"));
        w1.start();
        w2.start();
        w3.start();
    }
}
```

【习题 17-8】编写一个程序模拟警察的工作，按照警察公务的重要程度，设置线程的运行优先级。

【分析】该题主要考查线程的优先级。

【核心代码】

```
public class Police implements Runnable {
    private String name;

    public Police(String name) {
        this.name = name;
    }
    public void run() {
        System.out.println(name + "在执行公务");
    }

    public static void main(String[] args) {
        Thread p1 = new Thread(new Police("户籍警察"));
        Thread p2 = new Thread(new Police("国防警察"));
        Thread p3 = new Thread(new Police("缉毒警察"));
        p1.setPriority(2);
        p2.setPriority(10);
        p3.setPriority(8);
        p1.start();
        p2.start();
        p3.start();
    }
}
```

【习题 17-9】编写一个程序模拟士兵作战，如果负伤，就休整一段时间。

【分析】该题主要考查线程休眠。

【核心代码】

```
public class Soldier implements Runnable {
    private String name;

    public Soldier(String name) {
        this.name = name;
    }

    public void run() {
        for (int i = 0; i < 20; i++) {
            System.out.println(name + "作战中...");
            if (i == 13) {
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
                System.out.println(name + "负伤了，需要休整...");
                try {
                    Thread.sleep(3000);
                } catch (InterruptedException e) {
                }
            }
        }
    }

    public static void main(String[] args) {
        Thread s1 = new Thread(new Soldier("炮兵"));
        Thread s2 = new Thread(new Soldier("工兵"));
        Thread s3 = new Thread(new Soldier("特种兵"));
        s1.start();
        s2.start();
        s3.start();
    }
}
```

【习题 17-10】编写一个时钟程序。

【分析】该题主要考查对线程各个知识点的综合使用。

【核心代码】

```
public class DigitalClock extends JFrame implements Runnable {
    private Thread time;    // 时钟线程
    private JLabel clock;   // 用于显示时间
    private SimpleDateFormat format = new SimpleDateFormat("HH:mm:ss");
                            // 格式化时间

    public DigitalClock() {
        init();             // 初始化窗口
        start();            // 启动线程
    }

    public void init() {
        this.setTitle("Digital Clock");
        this.setSize(200, 100);
        this.setLocationRelativeTo(null);
        this.add(createLabel());
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setVisible(true);
    }

    public void start() {
        if (time == null) {
            time = new Thread(this);
            time.start();
        }
    }

    public void stop() {
        time = null;
    }

    public JLabel createLabel() {
        clock = new JLabel(format.format(new Date()), JLabel.CENTER);
                                                    // 创建一个JLabel，显示时间
        clock.setFont(new Font("Times New Roman", Font.BOLD, 40));  // 设置字体
        clock.setForeground(Color.BLUE);        // 设置颜色
        return clock;
    }
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
    public void run() {
        while (time != null) {
            try{
                Thread.sleep(1000);                          // 线程睡眠 1s
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            clock.setText(format.format(new Date()));  // 重新设置时间
        }
    }

    public static void main(String[] args) {
        new DigitalClock();                                  // 启动程序
        long l = 678756;
    }
}
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第 18 章　正则表达式

如果需要对字符串的内容进行分析匹配时，正则表达式是一个不错的选择。在 Java 中为正则表达式提供了比较丰富的类库，而且大大简化了使用正则表达式进行开发的工作。本章将主要介绍正则表达式。



18.1　正则表达式语法

所谓正则表达式，就是一个特殊的字符串，使用该字符串可以表示一种或者几种不同的匹配模式。正则表达式的基本语法如下。

- 普通字符：一个普通字符在表达式中只匹配与之相同的一个字母，例如，“k”在对字符串“ke”进行匹配时，匹配到的内容是字符串中的字符“k”。
- \r：表示回车键。
- \n：表示换行符。
- \t：表示制表符。
- \f：表示换页符。
- *：零次或多次匹配前面的字符或子表达式，等效于 {0,}。例如，zo* 与“z”和“zoo”匹配。
- +：一次或多次匹配前面的字符或子表达式，等效于 {1,}。例如，zo+ 与“zo”和“zoo”匹配，但与“z”不匹配。
- ?：零次或一次匹配前面的字符或子表达式，等效于 {0,1}。例如，当 ?紧随任何其他限定符（*、+、?、{n}、{n,} 或 {n,m}）之后时，匹配模式是非贪婪的。非贪婪模式匹配搜索到的、尽可能少的字符串，而默认的贪婪模式匹配搜索到的、尽可能多的字符串。
- ^：匹配搜索字符串开始的位置。如果标志中包括 m（多行搜索）字符，^ 还将匹配 \n 或 \r 后面的位置。
- $：匹配搜索字符串结尾的位置。如果标志中包括 m（多行搜索）字符，$ 还将匹配 \n 或 \r 前面的位置。 \d{3}$ 与搜索字符串结尾处的 3 个数字匹配。
- [] ：标记括号表达式的开始和结尾。[1-4] 与“1”、“2”、“3”或“4”匹配。[^aAeEiIoOuU] 与任何非元音字符匹配。
- {}：标记限定符表达式的开始和结尾。 a{2,3} 与“aa”和“aaa”匹配。
- () ：标记子表达式的开始和结尾。可以保存子表达式以备将来之用。 A(\d) 与“A0”～“A9”匹配。保存该数字以备将来之用。
- |：指示在两个或多个项之间进行选择。 zlfood 与“z”或“food”匹配，(zlf)ood 与“zood”或“food”匹配。
- /：表示 JavaScript 中的文本正则表达式模式的开始或结尾。在第二个“/”后添加单字符标志可以指定搜索行为。
- \：将下一字符标记为特殊字符、文本、反向引用或八进制转义符。 \n 与换行符匹配，\(与“(”匹配，\\ 与“\”匹配。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



- \b：与一个字边界匹配，即字与空格间的位置。 er\b 与“never”中的“er”匹配，但与“verb”中的“er”不匹配。
- \B：非边界字匹配。 er\B 与“verb”中的“er”匹配，但与“never”中的“er”不匹配。
- \d：数字字符匹配，等效于 [0-9]。 在搜索字符串“12 345”中，\d{2} 与“12”和“34”匹配，\d 与“1”、“2”、“3”、“4”和“5”匹配。
- \D：非数字字符匹配，等效于 [^0-9]。 \D+ 与“abc123 def”中的“abc”和“def”匹配。
- \w：与以下任意字符匹配：A～Z、a～z、0～9 和下画线，等效于 [A-Z a-z 0-9_]。在搜索字符串“The quick brown fox…”中，\w+ 与“The”、“quick”、“brown”和“fox”匹配。
- \W：与除 A～Z、a～z、0～9 和下画线以外的任意字符匹配，等效于 [^A-Z a-z 0-9_]。在搜索字符串“The quick brown fox…”中，\W+ 与“…”和所有的空格匹配。
- [xyz]：字符集，与任何一个指定字符匹配。 [abc] 与“plain”中的“a”匹配。
- [^xyz]：反向字符集，与未指定的任何字符匹配。 [^abc] 与“plain”中的“p”、“l”、“i”和“n”匹配。
- [a-z]：字符范围，匹配指定范围内的任何字符。 [a-z] 与“a”～“z”范围内的任何小写字母字符匹配。
- [^a-z]：反向字符范围，与不在指定范围内的任何字符匹配。[^a-z] 与不在范围“a”“z”内的任何字符匹配。
- {n}：正好匹配 n 次，其中，n 是非负整数。 o{2} 与“Bob”中的“o”不匹配，但与“food”中的两个“o”匹配。
- {n,}：至少匹配 n 次，其中 n 是非负整数。* 与 {0,} 相等，+ 与 {1,} 相等。
- o{2,}：与“Bob”中的“o”不匹配，但与“foooood”中的所有“o”匹配。
- {n,m}： 匹配至少 n 次，至多 m 次。n 和 m 是非负整数，其中 n≤m。注意，逗号和数字之间不能有空格。
- (模式)：与模式匹配并保存匹配项。您可以从由 JavaScript 中的 exec Method 返回的数组元素中检索保存的匹配项。若要匹配括号字符 ()，请使用“\(”或者“\)”。
- (?:模式)：与模式匹配，但不保存匹配项。即不会存储匹配项以备将来之用。这对于用“or”字符（|）组合模式部件的情况很有用。
- (?=模式)：积极的预测先行。找到一个匹配项后，将在匹配文本之前开始搜索下一个匹配项，不会保存匹配项以备将来之用。
- (?!模式)：消极的预测先行。匹配与模式不匹配的搜索字符串，找到一个匹配项后，将在匹配文本之前开始搜索下一个匹配项，不会保存匹配项以备将来之用。例如，\b(?!th)\w+\b 与不以“th”开头的单词匹配。
- \cx：：匹配 x 指示的控制字符，x 的值必须在 A～Z 或 a～z 范围内。如果不是这样，则假定 c 就是文本“c”字符本身。 \cM 与 Ctrl+M 或一个回车符匹配。
- \xn：匹配 n，此处的 n 是一个十六进制转义码。十六进制转义码必须正好是两位数长，允许在正则表达式中使用 ASCII 代码。
- \num：匹配 num，此处的 num 是一个正整数。这是对已保存的匹配项的引用。 (.)\1 与两个连续的相同字符匹配。
- \n：表示一个八进制转义码或反向引用。如果 \n 前面至少有 n 个捕获子表达式，那

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

么 n 是反向引用。否则，如果 n 是八进制数 (0～7)，那么 n 是八进制转义码。

- \nm：表示一个八进制转义码或反向引用。如果 \nm 前面至少有 nm 个捕获子表达式，那么 nm 是反向引用。如果 \nm 前面至少有 n 个捕获子表达式，则 n 是反向引用，后面跟有文本 m。如果上述情况都不存在，当 n 和 m 是八进制数 (0～7) 时，\nm 匹配八进制转义码 nm。
- \nml：当 n 是八进制数 (0～3)，m 和 l 是八进制数 (0～7) 时，匹配八进制转义码 nml。
- \un：匹配 n，其中 n 是以四位十六进制数表示的 Unicode 字符。 \u00A9 与版权符号 (©) 匹配。



18.2 正则表达式的使用

在 18.1 节中对正则表达式的语法做了详细的介绍，本节将主要介绍正则表达式的具体使用方法。在 Java 的标准库中提供了 Pattern 类和 Matcher 类，下面主要介绍这两个类。

18.2.1 Pattern 类

Pattern 类的对象表示通过编译的正则表达式。利用该对象可以和任意字符串进行模式的匹配。该类的对象需要调用 Pattern 类提供的静态工厂方法来调用，因为它所提供的构造方法是私有的。Pattern 中的静态工厂方法如下。

- public static Pattern compile(String regex)：其中的参数 regex 表示指定的用字符串表示的正则表达式，该方法会将指定的正则表达式编译成 Pattern 对象返回。
- public static Pattern compile(String regex,int flags)：参数 regex 表示指定的用字符串表示的正则表达式，第二个参数 flags 则表示指定的匹配标识，一般会使用常量表示。

在 Pattern 类中也提供了一些常用的标识常量。

- static int CANON_EQ：该常量表示启用规范等价。
- static int CASE_INSENSITIVE：该常量表示启用不区分大小写的匹配。
- static int COMMENTS：该常量表示模式中允许空白和注释。
- static int DOTALL ：该常量表示启用 dotall 模式。
- static int LITERAL：该常量表示启用模式的字面值解析。
- static int MULTILINE：该常量表示启用多行模式。
- static int UNICODE_CASE：该常量表示启用 Unicode 感知的大小写折叠。
- static int UNIX_LINES：该常量表示启用 UNIX 行模式。

以上是 Pattern 类中比较常用的标识常量，这些常量可以组合进行使用，如果需要组合使用，只要将需要组合的常量相加就可以了，例如以下代码：

```
Pattern=Pattern.compile("(ab)+",Pattern.CASE_INSENSITIVE+Pattern.UNICODE_CASE)
```

生成了 Pattern 对象之后，就可以使用对象中的方法进行相应的操作。在该类中，也提供了如下一些常用的方法。

- static Pattern compile(String regex)：该方法的作用是将给定的正则表达式编译到模式中。
- static Pattern compile(String regex, int flags)：该方法的作用是将给定的正则表达式编译到具有给定标志的模式中。
- int flags()：返回此模式的匹配标志。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



- Matcher matcher(CharSequence input)：该方法的作用是创建匹配给定输入与此模式的匹配器。
- static boolean matches(String regex, CharSequence input)：该方法的作用是编译给定正则表达式，并尝试将给定输入与其匹配。
- String pattern()：该方法的作用是返回在其中编译过此模式的正则表达式。
- static String quote(String s)：该方法的作用是返回指定 String 的字面值模式 String。
- String[] split(CharSequence input)：该方法的作用是围绕此模式的匹配拆分给定输入序列。
- String[] split(CharSequence input, int limit)：该方法的作用是围绕此模式的匹配拆分给定输入序列。
- String toString()：该方法的作用是返回此模式的字符串表示形式。

通过以上方法，就可以对字符串或者是字符序列进行匹配操作。

18.2.2　Matcher 类

进行字符串的模式匹配时，除了需要使用到 Pattern 对象之外，通常还需要 Matcher 类的对象，该对象表示将要进行模式匹配的字符串或者是字符序列。Matcher 和 Pattern 的对象一样不可以通过调用构造器来获得，一般是通过 Pattern 的 Matcher 方法来获得。通过 Pattern 生成了 Matcher 对象之后，就可以调用 Matcher 的方法进行字符串的匹配或者是替换的工作。Matcher 类的常用方法如下：

- Matcher appendReplacement(StringBuffer sb, String replacement)：该方法的作用是实现非终端添加和替换步骤。
- StringBuffer appendTail(StringBuffer sb)：该方法的作用是实现终端添加和替换步骤。
- int end()：该方法的作用是返回最后匹配字符之后的偏移量。
- int end(int group)：该方法的作用是返回在以前匹配操作期间，由给定组所捕获子序列的最后字符之后的偏移量。
- boolean find()：该方法的作用是尝试查找与该模式匹配的输入序列的下一个子序列。
- boolean find(int start)：该方法的作用是重置此匹配器，然后尝试查找匹配该模式，从指定索引开始的输入序列的下一个子序列。
- String group()：该方法的作用是返回由以前匹配操作所匹配的输入子序列。
- String group(int group)：该方法的作用是返回在以前匹配操作期间由给定组捕获的输入子序列。
- int groupCount()：该方法的作用是返回此匹配器模式中的捕获组数。
- boolean hasAnchoringBounds()：该方法的作用是查询此匹配器区域界限的定位。
- boolean hasTransparentBounds()：该方法的作用是查询此匹配器区域边界的透明度。
- boolean hitEnd()：该方法的作用是如果匹配器执行的最后匹配操作中搜索引擎遇到输入结尾，则返回 true。
- boolean lookingAt()：该方法的作用是尝试将从区域开头开始的输入序列与该模式匹配。
- boolean matches()：该方法的作用是尝试将整个区域与模式匹配。
- Pattern pattern()：该方法的作用是返回由此匹配器解释的模式。
- static String quoteReplacement(String s) ：该方法的作用是返回指定 String 的字面值替换 String。
- Matcher region(int start, int end)：该方法的作用是设置此匹配器的区域限制。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

- int regionEnd()：该方法的作用是报告此匹配器区域的结束索引（不包括）。
- int regionStart()：该方法的作用是报告此匹配器区域的开始索引。
- String replaceAll(String replacement)：该方法的作用是替换模式与给定替换字符串相匹配的输入序列的每个子序列。
- String replaceFirst(String replacement)：该方法的作用是替换模式与给定替换字符串匹配的输入序列的第一个子序列。
- boolean requireEnd()：如果很多输入都可以将正匹配更改为负匹配，则返回 true。
- Matcher reset()：该方法的作用是重置匹配器。
- Matcher reset(CharSequence input)：该方法的作用是重置此具有新输入序列的匹配器。
- int start()：返回以前匹配的初始索引。
- int start(int group)：返回在以前的匹配操作期间由给定组所捕获的子序列的初始索引。
- MatchResult toMatchResult()：作为 MatchResult 返回此匹配器的匹配状态。
- String toString()：返回匹配器的字符串表示形式。
- Matcher useAnchoringBounds(boolean b)：该方法的作用是设置匹配器区域界限的定位。
- Matcher usePattern(Pattern newPattern)：该方法的作用是更改此 Matcher 用于查找匹配项的 Pattern。
- Matcher useTransparentBounds(boolean b)：该方法的作用是设置此匹配器区域边界的透明度。

18.2.3 综合运用

在前面分别介绍了 Pattern 和 Matcher 类的创建以及它们的常用方法，下面通过一个示例介绍 Pattern 和 Matcher 中的方法在实际中的运用。

【范例 18-1】代码 18-1 是一个使用正则表达式的示例。

示例代码 18-1

```
01  public class PattMatTest
02  {
03      public static void main(String[] args)
04      {
05          //定义一个正则表达式和被匹配的字符串
06          String patternStr="(0?[1-9]|1[0-2]):([0-5]\\d)([a|p]m)";
07      StringmatcherStr="现在的时间是 11:15am，这里的营业时间是每天的 8:30am 至 5:30pm";
08          //生成 Pattern 对象和 Matcher 对象
09          Pattern pa=Pattern.compile(patternStr);
10          Matcher ma=pa.matcher(matcherStr);
11          //创建一个目标字符串缓冲区
12          StringBuffer sb=new StringBuffer();
13          //定义一个计数器
14          int count=0;
15          //将初始字符串打印出来
16          System.out.println("初始字符串是：");
17          System.out.println(matcherStr);
18          //使用 while 循环进行匹配检查
19          while(ma.find())
20          {
21              //创建一个临时字符串的缓冲区
22              StringBuffer temp=new StringBuffer();
23              if(ma.group(3).equals("am"))
24              {
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
25                      //如果时间为 am，则进行替换
26                      //设置事件的格式
27                      temp.append(ma.group(1));
28                      temp.append(":");
29                      temp.append(ma.group(2));
30                  }
31                  else
32                  {
33                      //设置时间为 pm 的进行替换
34                      //对时间进行提取
35                      int time=Integer.parseInt(ma.group(1));
36                      time=time+12;
37                      //设置新的时间格式
38                      temp.append(time+":"+ma.group(2));
39                  }
40                  //进行替换并将结果放置到目标字符串缓冲区中
41                  ma.appendReplacement(sb, temp.toString());
42                  //打印每次替换的情况
43                  System.out.println("["+(++count)+"]将"+ma.group(0)+"替换为"+temp.
                    toString());
44              }
45              //将最后一次替换后所剩余的字符串添加到目标字符串缓冲区
46              ma.appendTail(sb);
47              //打印最后的结果
48              System.out.println("最后的结果是");
49              System.out.println(sb.toString());
50          }
51      }
```

【运行结果】程序运行结果如下。

```
初始字符串是:
现在的时间是 11:15am，这里的营业时间是每天的 8:30am 至 5:30pm
[1]将 11:15am 替换为 11:15
[2]将 8:30am 替换为 8:30
[3]将 5:30pm 替换为 17:30
最后的结果是
现在的时间是 11:15，这里的营业时间是每天的 8:30 至 17:30
```

【代码解析】以上程序中，首先要定义正则表达式和被匹配字符串，并创建 Pattern、Matcher 对象（第 06～10 行）。使用 while 循环对字符串进行匹配检查，并将匹配字符串替换（第 19～43 行）。最后打印出最终的替换结果（第 48～49 行）。



18.3　String 类中的正则表达式应用

在复杂的匹配中要进行替换操作，使用 Pattern 和 Matcher 类是比较好的，但是简单的情况就没有必要使用这两个类了。为了简化开发，Java 中的 String 类也提供了一些基于正则表达式的操作方法。

18.3.1　模式匹配检查

String 类中有一个 matches 方法，该方法可以检查字符串中是否有匹配指定的模式，但是这个匹配是针对整个字符串而言的，即只能进行整体的匹配，不可以只匹配一部分。本节将主要介绍该方法。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
public Boolean matches(String regex);
```

- 该方法中的参数 regex 是指定以字符串形式表示的正则表达式。
- 该方法返回类型是布尔类型，是表达式测试的结果。如果整个字符串完全匹配指定的模式，则会返回 true，否则返回 false。
- 该方法中的参数 regex 所指定的表达式如果无效，就会抛出 java.util.regex.PatternSyntaxException 异常。

【范例 18-2】代码 18-2 是一个使用 matches 示例。

示例代码 18-2

```
01  public class MatchesTest {
02      public static void main(String[] args) {
03          //创建用字符串表示的正则表达式
04          String paterStr="\\d{4}-\\d{2}-\\d{2}";
05          //创建两个待匹配的字符串
06          String s1="9999-66-88";
07          String s2="内容是 9999-66-88";
08          //对字符串 s1 进行匹配检查
09          if(s1.matches(paterStr))
10          {
11              //匹配成功的情况
12              System.out.println("恭喜你，字符串 s1 满足指定格式的匹配要求");
13          }
14          else
15          {
16              //匹配不成功的情况
17              System.out.println("测试失败，字符串 s1 未能满足指定格式的匹配要求");
18          }
19          if(s2.matches(paterStr))
20          {
21              //匹配成功的情况
22              System.out.println("恭喜你，字符串 s2 满足指定格式的匹配要求");
23          }
24          else
25          {
26              //匹配不成功的情况
27              System.out.println("测试失败，字符串 s2 未能满足指定格式的匹配要求");
28          }
29      }
30  }
```

【运行结果】程序运行结果如下。

```
恭喜你，字符串 s1 满足指定格式的匹配要求
测试失败，字符串 s2 未能满足指定格式的匹配要求
```

【代码解析】以上程序中，首先分别设置正则表达式和两个不同的被匹配的字符串（第 04～07 行）。然后分别使用 matches 方法对其进行匹配，并打印出匹配结果（第 09～27 行）。

18.3.2 查找替换

在使用 Word 编辑文档时，会使用到查找和替换功能，也就是将一个字符串查找出来，或者是将该字符串替换成另一个字符串。使用正则表达式也可以完成这项功能，该方法和 Matcher 的替换功能类似。

```
public String replaceAll(String regex,String replacement);
public String replaceFirst(String regex,String replacement);
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



- 以上方法中的参数 regex 是用来匹配的正则表达式字符串，参数 replacement 是用来进行替换的字符串。
- replaceAll：该方法是将字符串中与指定模式相匹配的所有子串全部替换为制定的字符串 repacement，并且将替换后的结果以新的字符串形式返回。
- replaceFirst：该方法是将字符串中与指定模式相匹配的第一个子串全部替换为指定的字符串 replacement，返回形式是新的字符串形式。

【范例 18-3】示例代码 18-3 是一个使用查找替换示例。

示例代码 18-3

```
01  public class ReplaceTest
02  {
03      public static void main(String[] args)
04      {
05          //创建用字符串表示的正则表达式以及待匹配的字符串。
06          String patternStr="\\b\\d*\\.\\d{3,}\\b";
07          String s1="123.5 123.56 123.5678 123.5600 123.0098";
08          //打印待匹配的初始内容
09          System.out.println("初始字符串内容是：");
10          System.out.println(s1);
11          //进行 replaceAll 全部匹配替换操作
12          String s2=s1.replaceAll(patternStr, "Error");
13          //打印 replaceAll 全部匹配替换后的结果
14          System.out.println("使用 replaceAll 方法后结果为：");
15          System.out.println(s2);
16          //使用 replaceFirst 第一个匹配替换操作
17          s2=s1.replaceFirst(patternStr, "Error");
18          //打印 replaceFirst 第一个匹配替换操作的结果
19          System.out.println("使用 replaceFirst 方法后结果为：");
20          System.out.println(s2);
21      }
22  }
```

【运行结果】程序运行结果如下。

```
初始字符串内容是：
123.5 123.56 123.5678 123.5600 123.0098
使用 replaceAll 方法后结果为：
123.5 123.56 Error Error Error
使用 replaceFirst 方法后结果为：
123.5 123.56 Error 123.5600 123.0098
```

【代码解析】以上程序中，首先分别设置正则表达式和被匹配字符串（第 06～07 行）。然后分别使用 replaceAll 和 replaceFirst 方法对其进行匹配，并打印出匹配结果（第 12～20 行）。

18.3.3　字符串分析

在实际开发中，有时候需要将字符串进行拆分，拆分成若干个子字符串，我们可以使用 String 中的 split 方法来拆分字符串。

```
public String[] split(String regex);
public String[] split(String regex,int limit);
```

- 以上方法中的参数 regex 是用来作为分隔符的正则表达式，是字符串类型。第二个参数 limit 是用来限制拆分的份数，是整数类型。
- 第一种方法是按照指定的份数对字符串进行拆分。第二种方法是按照指定的份数对字

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

符串进行拆分。两个方法的返回类型都是字符串数组。

- 如果 limit 为正数，则该方法会将字符串拆分成 limit-1 份；如果 limit 为负数，则将尽可能多地进行拆分；如果 limit 为 0，则尽可能多地进行拆分，如果结尾为空字符串，将会被抛弃。

【范例 18-4】示例代码 18-4 是一个使用拆分示例。

示例代码 18-4

```
01  public class SplitTest
02  {
03      public static void main(String[] args)
04      {
05          //创建字符串表示的正则表达式和被匹配的字符串
06          String patternStr=";|,";
07          String  str="张三;李四,王五";
08          //打印出待拆分的字符串内容
09          System.out.println("需要拆分的字符串内容是：");
10          System.out.println(str);
11          //将字符串拆分为默认的份数
12          System.out.println("将字符串按照默认进行切割");
13          String[] arrStr=str.split(patternStr);
14          //打印拆分后的结果
15          for(int i=0;i<arrStr.length;i++)
16          {
17              System.out.println("arrStr["+i+"]=<"+arrStr[i]+">");
18          }
19          //将字符串拆分成 3 份
20          System.out.println("\n 将字符串拆分成 3 份");
21          arrStr=str.split(patternStr,3);
22          for(int i=0;i<arrStr.length;i++)
23          {
24              System.out.println("arrStr["+i+"]=<"+arrStr[i]+">");
25          }
26          //将字符串拆分成-2 份
27          System.out.println("\n 将字符串拆分成-2 份");
28          arrStr=str.split(patternStr,-2);
29          for(int i=0;i<arrStr.length;i++)
30          {
31              System.out.println("arrStr["+i+"]=<"+arrStr[i]+">");
32          }
33          //将字符串拆分成 0 份
34          System.out.println("\n 将字符串拆分成 0 份");
35          arrStr=str.split(patternStr,0);
36          for(int i=0;i<arrStr.length;i++)
37          {
38              System.out.println("arrStr["+i+"]=<"+arrStr[i]+">");
39          }
40      }
41  }
```

【运行结果】程序运行结果如下。

```
需要拆分的字符串内容是：
张三;李四,王五
将字符串按照默认值进行切割
arrStr[0]=<张三>arrStr[1]=<李四>arrStr[2]=<王五>
将字符串拆分成 3 份
arrStr[0]=<张三>
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
arrStr[1]=<李四>
arrStr[2]=<王五>

将字符串拆分成-2 份
arrStr[0]=<张三>
arrStr[1]=<李四>
arrStr[2]=<王五>

将字符串拆分成 0 份
arrStr[0]=<张三>
arrStr[1]=<李四>
arrStr[2]=<王五>
```

【代码解析】以上程序中，首先创建表示的正则表达式和被匹配的字符串（第 06、07 行）。然后分别以默认、拆分 3 份、拆分–2 份、拆分 0 份的设置来对被匹配字符串进行拆分，并打印出拆分结果（第 12～39 行）。



18.4　习题

【习题 18-1】编写一个程序验证输入的号码是否是手机号码。

【分析】该题主要考查正则表达式的语法。

【核心代码】

```
public class PhoneNumber {
    public static void main(String[] args) {
        Scanner reader = new Scanner(System.in); // 扫描输入文本
        System.out.println("请输入号码: ");
        String n = reader.next();
        String r = "^1[3|4|5|8][\\d]\\d{8}$";
        Pattern p = Pattern.compile(r);
        boolean b = p.matcher(n).matches();
        if (b) {
            System.out.println(n + " 是一个手机号码");
        } else {
            System.out.println(n + " 不是一个手机号码");
        }
    }
}
```

【习题 18-2】编写一个程序提取文本中的 E-mail 地址。

【分析】该题主要考查正则表达式的语法。

【核心代码】

```
public class EmailAddress {
    public static void main(String[] args) {
        String txt = "如果需要对字符串的内容进行分析匹配时，abc@aa.com abddatt@d.com 大大
地简化了使用正则表达式进行开发的工作";
        String r = "\\w+([-+.]\\w+)*@\\w+([-.]\\w+)*\\.\\w+([-.]\\w+)*";
                                                        // 此正则式值匹配字母
        Pattern p = Pattern.compile(r);
        Matcher m = p.matcher(txt);
        while (m.find()) {
            System.out.println(m.group(0));
        }
    }
}
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

【习题 18-3】编写一个程序验证输入的字符串是否是 URL 地址。

【分析】该题主要考查正则表达式的语法。

【核心代码】

```
public class URLAddress {
    public static void main(String[] args) {
        String s = "http://www.baidu.com/";
        String r = "^http://([\\w-]+\\.)+[\\w-]+(/[./?%&=]*)?$";
        Pattern p = Pattern.compile(r);
        boolean b = p.matcher(s).matches();
        if (b) {
            System.out.println(s + " 是一个 URL 地址");
        } else {
            System.out.println(s + " 不是一个 URL 地址");
        }
    }
}
```

【习题 18-4】编写一个程序验证输入的一段字母是否是 HTML 标记。

【分析】该题主要考查正则表达式的语法。

【核心代码】

```
public class HTMLTag {
    public static void main(String[] args) {
        String s = "<font color=#008000 size=-1>e.baidu.com</font>";
        String r = "<(.*)>.*</\\1|(.*)>|<(.*) />"; // 此正则表达式值匹配字母
        Pattern p = Pattern.compile(r);
        System.out.println(p.matcher(s).matches());
    }
}
```

【习题 18-5】编写一个程序查找给定文本中第一个符合正则表达式的字符串。

【分析】该题主要考查正则表达式的语法和字符串的查找。

【核心代码】

```
public class FindFirst {
    public static void main(String[] args) {
        String s = "荷叶生时春恨生,荷叶枯时秋恨成。深知身在情常在,怅望江头江水声。";
        String r = "(.*。)(.*。)";
        Pattern p = Pattern.compile(r);
        Matcher m = p.matcher(s);
        if (m.find()) {
            System.out.println(m.group(1)); //打印出前一句
        }
    }
}
```

【习题 18-6】编写一个程序查找给定文本中符合正则表达式的所有字符串。

【分析】该题主要考查正则表达式的语法和字符串的查找。

【核心代码】

```
public class FindAll {
    public static void main(String[] args) {
        String s = "If I were to fall in love, It would have to be with you. Your
                    eyes, your smile, The way you laugh, The things you say and do.
                    Take me to the places, My heart never knew. So, if I were to fall
                    in love, It would have to be with you. ";
        String r = "([^\\.]*\\.)";
        Pattern p = Pattern.compile(r);
        Matcher m = p.matcher(s);
        while (m.find()) {
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



```
            System.out.println(m.group(0).trim());
        }
    }
}
```

【习题 18-7】编写一个程序替换文本中首次出现的符合正则表达式的字符串。

【分析】该题主要考查正则表达式的语法和字符串的替换。

【核心代码】

```
public class ReplaceFirst {
    public static void main(String[] args) {
        String s = "荷叶生时春恨生,荷叶枯时秋恨成。深知身在情常在,怅望江头江水声。";
        String r = "\\,";
        System.out.println(s.replaceFirst(r, ", "));
    }
}
```

【习题 18-8】编写一个程序替换文本中所有的符合正则表达式的字符串。

【分析】该题主要考查正则表达式的语法和字符串的替换。

【核心代码】

```
public class ReplaceAll {
    public static void main(String[] args) {
        String s = "荷叶生时春恨生,荷叶枯时秋恨成。深知身在情长在,怅望江头江水声。";
        String r = "\\,";
        System.out.println(s.replaceAll(r, ", "));
    }
}
```

【习题 18-9】编写一个程序将字符串按给定正则表达式进行拆分。

【分析】该题主要考查正则表达式的语法和字符串的拆分。

【核心代码】

```
public class SplitString {
    public static void main(String[] args) {
        String s = "荷叶生时春恨生,荷叶枯时秋恨成。深知身在情常在,怅望江头江水声。";
        String r = ",|。";
        String[] ss = s.split(r);
        for (String a : ss) {
            System.out.println(a);
        }
    }
}
```

【习题 18-10】编写一个程序将字符串按给定正则表达式拆分成指定的份数。

【分析】该题主要考查正则表达式的语法和字符串的拆分。

【核心代码】

```
public class SplitStringByNum {
    public static void main(String[] args) {
        String s = "荷叶生时春恨生,荷叶枯时秋恨成。深知身在情常在,怅望江头江水声。";
        String r = ",|。";
        String[] ss = s.split(r, 3);
        for (String a : ss) {
            System.out.println(a);
        }
    }
}
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第 19 章　封　装　类

Java 语言中为每一种基本的数据类型都提供了一个相对应的封装类，例如，Integer 类所对应的 int 型、Float 类所对应的 float 型等。这样就大大加快了程序的处理速度，提高了程序的性能。Java 中的封装类主要有 Double、Float、Long、Integer、Short、Byte、Character 和 Boolean 等。本章将对这些类进行介绍。



19.1　常用的数据类型包装类

19.1.1　Integer 类

Integer 是基本数据类型的包装类，Integer 类在对象中包装了一个基本类型 int 的值，每个 Integer 类型的对象包含一个 int 类型的字段。该类的构造方法如下：

- Integer(int value)：构造一个新分配的 Integer 对象，它表示指定的 int 值。
- Integer(String s)：构造一个新分配的 Integer 对象，它表示指定的字符串所代表的 int 值。

该类的一些字段如下：

- MAX_VALUE：保持 int 类型的最大值的常量可取的值为 231～1。
- MIN_VALUE：保持 int 类型的最小值的常量可取的值为 –231。
- SIZE：以二进制补码形式表示 int 值的位数。
- TYPE：表示基本类型 int 的 Class 实例。

该类还提供了一系列的方法，常用方法如表 19-1 所示。

表 19-1　Integer 的常用方法

方 法 名	说　明
public Integer(int value)	构造一个新的 Integer 对象，用来表示参数的 int 值
public Integer(String s)throws NumberFormatException	构造一个新的 Integer 对象，用来表示 String 参数所指定的 int 值。当无法转换的时候会抛出 NumberFormatException 异常
public static String toHexString(i)	以十六进制无符号整数形式返回指定参数的十六进制表示形式
public static String toOctalString(itn i)	以八进制无符号形式返回参数所表示的 int 值的字符串形式
public static String toBinaryString(int i)	以二进制无符号形式返回参数表示的 int 值的字符串形式
public static String toString(int i)	返回一个表示参数 i 的 String 对象
public static int parseInt(String s)trows NumberFormatException	返回一个表示参数 s 的 int 值。常用于进行类型转换，当无法进行转换的时候会抛出 NumberFormatException 异常
public static Integer valueOf(int i)	返回一个 Integer 对象用来表示参数指定的 int 值
public byteValue()	以 byte 类型返回该 Integer 对象的值
public short shortValue()	返回此 Integer 对象的 short 值
public int intValue()	返回此 Integer 对象的 int 值
public long longValue()	返回此 Integer 对象的 long 值
public float floatValue()	返回此 Integer 对象的 float 值

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

续表

方　法　名	说　　明
public double doubleValue()	返回此 Integer 对象的 double 值
public static Integer decode(String s)	将 String 类型的 s 解码为 Integer 对象

下面的程序会使用这些字段和方法。

【范例 19-1】示例代码 19-1 演示 Integer 类的使用。

示例代码 19-1

```
01  public class IntegerTest
02  {
03          public static void main(String[] args)
04          {
05          //打印 Integer 的最大值
06          System.out.println("Integer 的最大值是："+Integer.MAX_VALUE);
07          String s=new String("1010");
08          //返回 s 的十进制数
09          System.out.println("s 的十进制数是："+Integer.getInteger(s, 2));
10          Integer i=new Integer("1234");
11          //比较
12          System.out.println("i 的比较："+i.compareTo(123));
13          int j=12345;
14          //转换为字符串
15          System.out.println("转换字符串结果为："+Integer.toString(j, 16));
16      }
17  }
```

【运行结果】程序的运行结果如下：

```
Integer 的最大值是：2147483647
s 的十进制数是：2
i 的比较：1
转换字符串结果为：3039
```

【代码解析】在以上程序中，分别演示了 Integer 类的具体应用（第 05～16 行）。

19.1.2　Double 类

Double 类是基本数据类型 double 的封装类，它的构造方法如下。

- Double(double value)：构造一个新分配的 Double 对象，它表示基本的 double 参数。
- Double(String s)：构造一个新分配的 Double 对象，表示用字符串表示的 double 类型的浮点值。

每个 Double 类型的对象都包含一个 double 类型的字段。该类的一些重要字段如表 19-2 所示。

表 19-2　Double 的重要字段

字　段　名	说　　明
public static final double MAX_VALUE	保持 double 类型的最大的正有限值的常量
public static final double MIN_VALUE	保持 double 类型数据的最小正非零值的常量
public static final int SIZE	用于表示 double 值的位数
public static final Class<Double>Type	返回表示基本类型 double 的类
public static final double POSITIVE_INFINITY	保存 double 类型的正无穷大的常量
public static final double NEGATIVE_INFINITY	保存 double 类型的负无穷大的常量

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

下面的程序演示了这些常量。

【范例 19-2】示例代码 19-2 演示 Double 类的使用。

示例代码 19-2

```
01  public class DoubleTest
02  {
03      public static void main(String[] args)
04      {
05          Double dd=new Double(56);
06          //打印出 Double 类的一些常量
07          System.out.println("Double 的最大值是："+dd.MAX_VALUE);
08          System.out.println("Double 的最小值是："+dd.MIN_VALUE);
09          System.out.println("Double 的类型是："+dd.TYPE);
10          System.out.println("Double 的数据大小是："+dd.SIZE);
11          System.out.println("Double 的正无穷是："+dd.POSITIVE_INFINITY);
12          System.out.println("Double 的负无穷是："+dd.NEGATIVE_INFINITY);
13      }
14  }
```

【运行结果】程序的输出如下：

```
Double 的最大值是：1.7976931348623157E308
Double 的最小值是：4.9E-324
Double 的类型是：double
Double 的数据大小是：64
Double 的正无穷是：Infinity
Double 的负无穷是：-Infinity
```

Double 类还提供了一系列的方法，如表 19-3 所示。

表 19-3　Double 中的方法

方 法 名	说　明
public Double(double value)	构造一个新的 Double 对象，用来表示 value
public Double(String s)throws NumberFormatException	构造一个 Double 对象，用来表示 s 指定的浮点值。注意，该方法可能会抛出 NumberFormatException 异常
public static String toString(double d)	返回 double 参数的字符串表示形式
public static String toHexString(double d)	返回 double 参数的十六进制字符串表示形式。详细的信息请读者自行查阅 API 了解相关信息
public static Double valueOf(String s)throws NumberFormatException	返回表示字符串 s 的 Double 对象
public static Double vlueOf(double d)	返回参数值 d 的 Double 实例。这种形式比用构造函数来构造 Double 对象效率要高
public static double parseDouble(String s)throws NumberFormatException	返回一个新的 double 值，通过该值表示参数 s。注意，该方法可能会抛出 NumberFormatException 异常
public static boolean isNaN(double v)	如果指定的参数 v 为一个非数字（NaN）值，则返回 true，否则返回 false
public static boolean isInfinite(double v)	如果指定数字的大小是无穷大，则返回 true；否则，返回 false
public byte byteValue()	返回此 Double 对象的 byte 值
public short shortValue()	返回此 Double 对象的 short 值
public int intValue()	返回此 Double 对象的 int 值
public long longValue()	返回此 Double 对象的 long 值
public double doubleValue()	返回此 Double 对象的 double 值

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



以上方法主要是运用于 Double 和其他数据类型之间的转换。其中需要注意的是，在 String 向 Double 类型转换的过程中，有可能会抛出 NumberFormatException 异常。

19.1.3　Number 类

抽象类 Number 是 Byte、Double、Float、Integer、Long 和 Short 类的超类。Number 的子类必须提供将表示的数值转换为 byte、double、float、int、long 和 short 类型的方法。可以这样使用 Number 类：

```
Number n=new Double(0);
```

Number 类提供了一些方法，它的子类都实现了这些方法，如表 19-4 所示。

表 19-4　Number 方法

方 法 名	说　明
public byte byteValue()	以 byte 形式返回指定的数值
public short shorValue()	以 short 形式返回指定的数值
public abstract int intValue()	以 int 形式返回指定的数值。这是一个抽象方法，子类要实现这个方法
public abstract long longValue()	以 long 形式返回指定的数值。这是一个抽象方法，子类要实现这个方法
public abstract float floatValue()	以 float 形式返回指定的数值。这是一个抽象方法，子类要实现这个方法
public abstract double doubleValue()	以 double 形式返回指定的数值。这是一个抽象方法，子类要实现这个方法

19.1.4　Float 类

Float 类在对象中包装了一个 float 基本类型的值，每个 Float 类型的对象包含一个 float 类型的字段。Float 类的一些字段描述了 float 数据类型的一些属性，如表 19-5 所示。

表 19.5　float 字段

字 段 名	说　明
public static final float MAX_VALUE	保存 float 类型的最大正有限大值的常量
public static final float MIN_VALUE	保存 float 类型数据的最小正非零值的常量
public static final int SIZE	表示一个 float 值所使用的位数
public static final Class<Float> TYPE	表示 float 基本类型的 Class 实例
public static final float POSITIVE_INFINITY	保存 float 类型的正无穷大的常量
public static final float NEGATIVE_INFINITY	保存 float 类型的负无穷大值的常量

此外，Float 类还提供了一系列的方法，如表 19-6 所示。

表 19-6　float 方法

方 法 名	说　明
public Float(float value)	构造一个新 Float 对象，表示 float 参数
public Float(double value)	构造一个新 Float 对象，表示 double 参数
public Float(String value)	构造一个新 Float 对象，是 String 参数表示的值
public static String toString()	返回 float 参数的字符串形式
public static String toHexString(float f)	返回 float 参数的十六进制字符串表示形式
public static Float valueOf(String s)throws NumberFormatException	返回字符串 s 表示的 float 值的 Float 对象。注意，当无法转换的时候可能会抛出 NumberFormatException 异常
public static Float valueOf(float f)	返回一个表示参数 float 值的 Float 对象

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

续表

方 法 名	说 明
public static float pareseFloat(String s) throws NumberFormatException	返回 String 类型参数 s 代表的 float 值
public static boolean isNaN(float v)	如果参数是一个非数字值，则返回 true，否则返回 false
public static boolean isInfinite(float v)	如果指定数字的大小是无穷大，则返回 true，否则返回 false
public byte byteValue()	返回此 Float 对象的 byte 值
public shortValue()	返回此 Float 对象的 short 值
public int intValue()	返回这个 Float 对象对应的 int 值
public long longValue()	返回这个 Float 对象对应的 long 值
public float intValue()	返回这个 Float 对象对应的 float 值
public double doubleValue()	返回这个 Float 对象的 double 值

下面的示例是演示这些字段的值和这些方法的应用。

【范例 19-3】示例代码 19-3 演示 Float 类的使用。

示例代码 19-3

```
01  public class FloatDemo
02  {
03      public static void main(String[] args) {
04          //创建一个 Float 对象
05          Float f=new Float(20f);
06          //打印出 Float 类定义的常量
07          System.out.println("Float 最大值: "+f.MAX_VALUE);
08          System.out.println("Float 最小值: "+f.MIN_VALUE);
09          System.out.println("Float 位数: "+f.SIZE);
10          System.out.println("Float 正无穷大的常量:"+f.POSITIVE_INFINITY);
11          System.out.println("Float 负无穷大的常量:"+f.NEGATIVE_INFINITY);
12          System.out.println("f 的值: "+f.floatValue());
13          System.out.println("f 的十六进制形式"+f.toHexString(f));
14          System.out.println("f 的 double 值"+f.doubleValue());
15      }
16  }
```

【运行结果】程序运行结果如下：

```
Float 最大值: 3.4028235E38
Float 最小值: 1.4E-45
Float 位数: 32
Float 正无穷大的常量:Infinity
Float 负无穷大的常量:-Infinity
f 的值: 20.0
f 的十六进制形式 0x1.4p4
f 的 double 值 20.0
```

【代码解析】以上程序中，分别使用表 19-6 中的方法打印输出 float 相关的属性（第 07～14 行）。

19.1.5 Boolean 类

对于 Boolean 类型，Java 也提供了相应的封装类 Boolean 类。Boolean 类将基本类型为 boolean 的值包装在一个对象中。一个 Boolean 类型的对象只包含一个类型为 boolean 的字段，

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



该类包含三个字段，如表 19-7 所示。

表 19-7　Boolean 字段

字　段　名	说　　明
public static fianal Boolean TRUE	对应一个 Boolean 对象，这个对象表示 true
public static final Boolean FALSE	对应一个 Boolean 对象，这个对象表示 false
public static final Class<Boolean>TYPE	表示基本类型 Boolean 类型的 Class 对象

该类还提供了一些方法，如表 19-8 所示。

表 19-8　Boolean 中的方法

方　法　名	说　　明
public Boolean(boolean value)	分配一个表示 value 值的对象
public Boolean(String s)	当 s 在忽略大小写的情况下为“true”时，则产生一个表示 true 的对象，其他的情况产生一个“false”对象
public static boolean pareseBoolean(String s)	将字符串转换为 boolean 值。当 s 在忽略大小写的情况下为“true”时，则产生一个表示 true，其他的情况表示“false”
public boolean booleanValue()	返回此 Boolean 对象表示的 boolean 值
public static Boolean valueOf(boolean b)	返回一个表示参数 b 的 Boolean 对象
public static Boolean valueOf(String s)	返回 String 参数表示的 boolean 值的 Boolean 对象，根据 String 来判断，s 忽略大小写为“true”时，返回一个表示 true 的 Boolean 对象，否则返回一个 false 的 Boolean 对象
public static boolean pareseBoolean(String s)	将字符串转换为 boolean 类型，规则同上
public boolean booleanValue()	返回此 Boolean 对象表示的 boolean 值
public static Boolean valueOf(boolean b)	返回一个表示指定 boolean 值的 Boolean 对象
public static Boolean valueOf(String s)	返回一个 String 参数表示的 boolean 值的 Boolean 对象，转换规则同上

下面的程序演示了这些方法和字段的使用。

【范例 19-4】示例代码 19-4 演示 Boolean 类的使用。

示例代码 19-4

```
public class BooleanTeat
{
01      public static void main(String[] args)
02      {
03          //通过两个不同的构造方法来构造对象
04          Boolean b1=Boolean.valueOf(true);
05          Boolean b2=Boolean.valueOf("true");
06          Boolean b3=Boolean.valueOf("a");
07          //用构造函数构造对象
08          Boolean b4=new Boolean("true");
09          Boolean b5=new Boolean("a");
10          Boolean b6=new Boolean(true);
11          //打印出这些 Boolean 对象的值
12          System.out.println("b1 对象的值："+b1.booleanValue());
13          System.out.println("b2 对象的值："+b2.booleanValue());
14          System.out.println("b3 对象的值："+b3.booleanValue());
15          System.out.println("b4 对象的值："+b4.booleanValue());
16          System.out.println("b5 对象的值："+b5.booleanValue());
17          System.out.println("b6 对象的值："+b6.booleanValue());
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
18        }
19    }
```

【运行结果】程序运行的结果如下：

```
b1 对象的值：true
b2 对象的值：true
b3 对象的值：false
b4 对象的值：true
b5 对象的值：false
b6 对象的值：true
```

【代码解析】以上程序中，首先使用不同的构造方法来构造 Boolean 的对象（第 04～10 行），然后对各个对象的值打印输出（第 12～17 行）。

19.1.6 Byte 类

Byte 类是基本数据类型 byte 的封装类，一个 Byte 类型的对象只包含一个类型为 byte 的字段。Byte 类有一些字段如表 19-9 所示。

表 19-9 Byte 字段

字 段 名	说 明
public static final byte MAX_VALUE	一个常量用来保存 byte 类型可取的最大值 2^7-1
public static final byte MIN_VALUE	一个常量，保存 byte 类型可取的最小值，即-128
public static final int SIZE	用于以二进制补码形式表示 byte 值的位数

Byte 类还提供了一系列的方法，如表 19-10 所示。

表 19-10 Byte 方法

方 法 名	说 明
public Byte(byte value)	构造一个新的 Byte 对象，用以表示指定的 byte 值
public Byte(String s)thows NumberFormatException	构造一个新分配的 Byte 对象，以表示 string 参数表示的 byte 值。如果字符串转换错误会抛出 NumberFormatException 异常
public static String(byte b)	返回一个表示指定 byte 的 String 对象
public static Byte valueOf(byte b)	返回表示参数 byte 值的一个 Byte 对象
public static byte parseByte(String s)throws NumberFormatException	将 String 参数转换为 byte 类型。第一个字符可以用来表示符号，其他的都必须为数字，否则会抛出异常。
public static Byte valueOf(String s)throws NumberFormatException	返回一个 byte 值表示参数指定的 string。参数 s 除了第一位可以表示符号的负号外，其他的都必须为数字，否则会抛出异常
public static Byte decode(String s)throws NumberFormatException	将参数 s 解码为 Byte
public byte byteValue()	返回此 Byte 对象的 byte 值
public short shortValue()	返回此 Byte 对象的 float 值
public int intValue()	返回此 Byte 对象的 int 值
public long longValue()	返回此 Byte 对象的 long 值
public float floatValue()	返回此 Byte 对象的 float 值
public double doubleValue()	返回此 Byte 对象的 double 值

这些方法主要用于各种数据类型的转化，包括 String 类型。需要注意的一点是，当把一个 String 类型构造成 Byte 类型时，可能会发生 NumberFormatException 异常，这时表示转化失败。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



19.2　StringBuffer 类

StringBuffer 是 Java 中另一种对字符串进行操作的类。StringBuffer 中解决了 String 类中不可变的缺点，也解决了字符串连接不方便的问题。本节将主要介绍字符串缓冲类 StringBuffer。

19.2.1　StringBuffer 简介

StringBuffer 也是对字符串进行操作的类，它和 String 是对等的。如果需要对字符串进行大量的修改时，就要使用到 StringBuffer 类。StringBuffer 是可以改变的字符序列，它的常用操作就是将字符串插入到字符序列中和追加到字符序列的尾部。StringBuffer 是线程安全的，缓冲序列可以自动对多线程的操作进行同步处理，但是会影响到性能。该类的构造方法如表 19-11 所示。

表 19-11　StringBuffer 构造方法

方　法　名	说　　明
public StringBuffer()	构造一个不带字符的字符串缓冲区，初始容量为 16 个字符
public StringBuffer(int capacity)	构造一个不带字符的，但是具有指定容量的字符串缓冲区
public StringBuffer(String str)	构造一个字符串缓冲区，并将缓冲区的内容指定为字符串 str 的内容。该字符串缓冲区的容量为 str 的长度再加上 16
public StringBuffer(CharSequence seq)	构造一个字符串缓冲区，包含指定字符序列，该字符串缓冲区的容量为字符串序列 seq 的长度再加上 16

19.2.2　StringBuffer 类的方法

StringBuffer 提供了一系列的方法，如表 19-12 所示。

表 19-12　StringBuffer 方法

方　法　名	说　　明
public StringBuffer append(String str)	append()方法是把 str 的内容追加到 StringBuffer 的末尾，可以在 StringBuffer 后追加的内容不仅仅有 String 类型的，也可以是 char、int、long 等类型
public int length()	返回 StringBuffer 的当前字符序列长度
public int capacity()	返回字符串缓冲区当前的容量，即可以最新插入的字符存储量。当插入的字符串长度大于这个容量的时候需要重新分配
public void ensureCapacity(int minimumCapacity)	确保容量至少等于 minimumCapacity 指定的值。容量小于这个值的时候，重新分配缓冲区
public char charAt(int index)	返回此字符序列中索引 index 处的字符，索引从 0 开始
public void setCharAt(int index,char ch)	用 ch 替换索引 index 处的字符
public StringBuffer deleteCharAt(int index)	删除索引 index 处的字符
public StringBuffer delete(int start,int end)	删除字符序列从 start 位置起到 end-1 处的字符，注意包括 start 位置的字符，但是不包括 end 位置处的字符
public StringBuffer replace(int start ,int end ,String str)	用参数指定的字符串 str 替换字符序列从 start 位置开始到 end-1 处结束的字符，注意，不包括 end 位置的字符
public String substring(int start)	返回一个 String 字符串，该字符串从字符序列的 start 位置开始一直到字符序列的末尾

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

续表

方 法 名	说 明
public int indexOf(str)	返回第一次出现指定子字符串的位置索引
public int lastIndexOf(String str)	返回在字符串序列中，最右边的出现指定子字符串 str 的索引位置
public int indexOf(String str,int fromIndex)	从指定的索引处开始，第一次出现指定子字符串的位置索引
public StringBuffer reverse()	将此字符序列取其翻转形式
public void setLength(int length)	该方法用于设置字符序列的长度。当对象所有的字符序列小于 length 时，会把它的长度改为 length，如果大于 length 时，取前 length 个字符保留

本节将会对这些方法进行介绍，有的方法会用简单的例子进行说明。

下面的程序演示了各类数据的 append()。

【范例 19-5】示例代码 19-5 演示 StringBuffer 类连接字符串。

示例代码 19-5

```
01  public class AppendTest
02  {
03      public static void main(String args[])
04      {
05      StringBuffer sb=new StringBuffer("Hello");  //创建一个 StringBuffer 对象 sb
06          sb.append('!');                         //在 sb 的尾部追加一系列元素
07          sb.append("\n");
08          sb.append(true);
09          sb.append("\n");
10          sb.append(20);
11          sb.append("\n");
12          sb.append("Hello!");
13          System.out.println(sb);                 //打印出结果
14      }
15  }
```

【运行结果】程序的运行结果如下：

```
Hello!
true
20
Hello!
```

【代码解析】以上程序中，首先创建一个 StringBuffer 对象（第 05 行）。然后根据不同的字符进行拆分（第 06～12 行）。最后将 StringBuffer 对象打印输出（第 13 行）。

下面是一个小程序，取得 StringBuffer 对象每个位置的字符，用大写形式替换它们，在程序中使用了这两个方法。

【范例 19-6】示例代码 19-6 演示 StringBuffer 类字符处理。

示例代码 19-6

```
01  public class StringBufferTest2
02  {
03      public static void main(String[] args)
04      {
05      StringBuffer sb=new StringBuffer("abcdefg");  //创建一个 StringBuffer 对象
06          System.out.println(sb);                   //打印出 sb 的值
07          //取得每个位置的字符，并用大写形式替换
08          for (int i = 0; i < sb.length(); i++)
09          {
10              char c=sb.charAt(i);
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
11                  System.out.println(c);
12                  c=(char) (c-32);                    //取得 c 的大写形式
13                  sb.setCharAt(i, c);                 //替换
14              }
15              System.out.println(sb);
16          }
17      }
```

【运行结果】程序的输出结果如下：

```
abcdefg
a
b
c
d
e
f
g
ABCDEFG
```

【代码解析】以上程序中，首先创建一个 StringBuffer 对象，并将其值打印输出（第 05、06 行）。使用 for 循环取得每一个位置的字符，并将该字符用大写的形式替换（第 08～14 行）。最后将替换后的结果打印输出（第 15 行）。

下面的程序是替换文字中的一段指定字符。

【范例 19-7】示例代码 19-7 演示 StringBuffer 类字符串处理。

示例代码 19-7

```
01  public class ReplaceTest
02  {
03      public static void main(String[] args)
04      {
05          //构造一个 StringBuffer 对象
06          StringBuffer sb=new StringBuffer
07          ("Tom is a good sttudent,Tom love his mother,Tom like paying football");
08          System.out.println(sb);          //打印出 StringBuffer 对象内容
09          int index=0;                     //用 index 来表示下次调用 indexOf 的位置
10          //需要用 index 和名字的长度 length 来对 indexOf 操作进行定位
11          int length="Tom".length();
12          String name="Mark";              //用 mark 来替换 tom
13          //循环语句，直到全部被替换完毕为止
14          while (sb.indexOf("Tom",index+1)>index)
15          {
16              index=s.indexOf("Tom");                          //取得位置
17              sb.replace(index, index+length, name);           //替换
18          }
19          System.out.println(sb);                              //打印出结果
20      }
21  }
```

【运行结果】下面是程序的运行结果：

```
Tom is a good sttudent,Tom love his mother,Tom like paying football
Mark is a good sttudent,Mark love his mother,Mark like paying football
```

19.3　日期操作类

日期是 Java 中比较复杂的内容。不同的国家，有不同的日期格式，日期的国际化、日期

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

和时间之间的转换问题都是非常复杂的。Java 中对日期的操作涉及 Date、DateFormat、Calendar 等类，下面就对经常使用的类进行介绍。

19.3.1 Date 类

Date 表示的是特定的时间，可以精确到毫秒，用 DateFormat 类来对日期进行格式化和分析日期字符串。Date 中的构造方法如下：

- Date()：分配 Date 对象，并初始化该对象，以表示分配它的时间。
- Date(long date)：分配 Date 对象并初始化该对象，以表示从标准的基准时间以来制定的毫秒数。

Date 中也有一些成员方法，如表 19-13 所示。

表 19-13 Date 方法

方 法 名	说 明
boolean after(Date when)	测试此日期是否在指定日期之后
boolean before(Date when)	测试此日期是否在指定日期之前
Object clone()	返回此对象的副本
int compareTo(Date anotherDate)	比较两个日期的顺序
boolean equals(Object obj)	比较两个日期的相等性
void setTime(long time)	设置此 Date 对象，以表示 1970 年 1 月 1 日 00:00:00 GMT 以后 time 毫秒的时间点
String toString()	把此 Date 对象转换为以下形式的 String：dow mon dd hh:mm:ss zzz yyyy，其中，dow 是一周中的某一天（Sun、Mon、Tue、Wed、Thu、Fri、Sat）
int hashCode()	返回此对象的哈希码值

下面的示例是使用 Date 类中的方法来获取今天的日期。

【范例 19-8】示例代码 19-8 是一个使用拆分示例。

示例代码 19-8

```
public class DateTest {

01      public static void main(String[] args) {
02          //获取系统当前时间
03          System.out.println("当前时间是："+System.currentTimeMillis());
04          //获取系统当前日期
05          Date date=new Date();
06          System.out.println("现在的时间是："+date.toString());
07          //获取从标准基准时间到现在的毫秒数
08          System.out.println("自 1970 年 1 月 1 日 0 时 0 分 0 秒开始至今经历的毫秒数：
                            "+date.getTime());
09      }
10  }
```

【运行结果】程序运行结果如下：

```
当前时间是：1277887841656
现在的时间是：Wed Jun 30 16:50:41 CST 2010
自 1970 年 1 月 1 日 0 时 0 分 0 秒开始至今经历的毫秒数：1277887841656
```

【代码解析】以上程序中，分别获取当前时间、日期和从基准时间到现在的毫秒数，并将其结果打印输出（第 03～08 行）。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



19.3.2　DateFormat 类

DateFormat 类是日期时间格式化子类的抽象类，它可以与语言无关的方法格式化，并进行分析日期或者时间。DateFormat 中定义了一些方法，如表 19-14 所示。

表 19-14　DateFormat 方法

方　法　名	说　　明
Object clone()	重写 Cloneable
boolean equals(Object obj)	重写 equals
String format(Date date)	将一个 Date 格式化为日期/时间字符串
abstract StringBuffer format(Date date, StringBuffer toAppendTo, FieldPosition fieldPosition)	将一个 Date 格式化为日期/时间字符串
StringBuffer format(Object obj, StringBuffer toAppendTo, FieldPosition fieldPosition)	重写 Format
static Locale[] getAvailableLocales()	返回所有语言环境的数组，此类的 get*Instance 方法可以为其返回已本地化的实例
Calendar getCalendar()	获取与此日期/时间格式器关联的日历
static DateFormat getDateInstance()	获取日期格式器，该格式器具有默认语言环境的默认格式化风格
static DateFormat getDateInstance(int style)	获取日期格式器，该格式器具有默认语言环境的给定格式化风格
static DateFormat getDateInstance(int style, Locale aLocale)	获取日期格式器，该格式器具有给定语言环境的给定格式化风格
static DateFormat getDateTimeInstance()	获取日期/时间格式器，该格式器具有默认语言环境的默认格式化风格
static DateFormat getDateTimeInstance(int dateStyle, int timeStyle)	获取日期/时间格式器，该格式器具有默认语言环境的给定日期和时间格式化风格
static DateFormat getDateTimeInstance(int dateStyle, int timeStyle, Locale aLocale)	获取日期/时间格式器，该格式器具有给定语言环境的给定格式化风格
static DateFormat getInstance()	获取为日期和时间使用 SHORT 风格的默认日期/时间格式器
NumberFormat getNumberFormat()	获取此日期/时间格式器用于格式化和解析时间的数字格式器
static DateFormat getTimeInstance()	获取时间格式器，该格式器具有默认语言环境的默认格式化风格
static DateFormat getTimeInstance(int style)	获取时间格式器，该格式器具有默认语言环境的给定格式化风格
static DateFormat getTimeInstance(int style, Locale aLocale)	获取时间格式器，该格式器具有给定语言环境的给定格式化风格
TimeZone getTimeZone()	获取时区
int hashCode()	重写 hashCode
boolean isLenient()	判断日期/时间解析是否为不严格的
Date parse(String source)	从给定字符串的开始解析文本，以生成一个日期
abstract Date parse(String source, Parse Position pos)	根据给定的解析位置开始解析日期/时间字符串
Object parseObject(String source, Parse Position pos)	解析字符串中的文本，以生成一个 Date
void setCalendar(Calendar newCalendar)	设置此日期格式所使用的日历
void setLenient(boolean lenient)	指定日期/时间解析是否不严格
void setNumberFormat(NumberFormat new NumberFormat)	允许用户设置数字格式器
void setTimeZone(TimeZone zone)	为此 DateFormat 对象的日历设置时区

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

下面是一个使用 SimpleDateFormate 类对日期进行格式化的示例。

【范例 19-9】示例代码 19-9 是一个使用拆分示例。

示例代码 19-9

```
01  public class DateFormatTest {
02
03      public static void main(String[] args) {
04
05      Date date=new Date();
06      DateFormat df1=DateFormat.getInstance();
07      DateFormat df2=new SimpleDateFormat("yyyy-MM-dd hh:mm:ss EE");
08      DateFormat df3=DateFormat.getDateInstance(DateFormat.FULL,
09                      Locale.CHINA);
10      DateFormat df4=new SimpleDateFormat("yyyy 年 MM 月 dd 日 hh 时 mm 分 ss 秒 EE",
11                      Locale.CHINA);
12      DateFormat df5=new SimpleDateFormat("yyyy-MM-dd hh:mm:ss EEEEEE",
13                      Locale.US);
14      DateFormat df6=new SimpleDateFormat("yyyy-MM-dd");
15      DateFormat df7=new SimpleDateFormat("yyyy 年 MM 月 dd 日");
16
17      System.out.println("按照 Java 的默认格式进行输出，默认的区域是："+
18                          df1.format(date));
19      System.out.println("按照指定格式 yyyy-MM-dd hh:mm:ss EE，系统默认区域是："+
20                          df2.format(date));
21      System.out.println("按照日期的 FULL 模式，区域设置为中文："+
22                          df3.format(date));
23      System.out.println("按照指定格式 yyyy 年 MM 月 dd 日 hh 时 mm 分 ss 秒 EE，区域
                            为中文："+
24              df4.format(date));
25      System.out.println("按照指定格式 yyyy-MM-dd hh:mm:ss EEEEEE，区域是美国："+
26                          df5.format(date));
27
28      }
29
30  }
```

【运行结果】程序运行结果如下：

```
按照 Java 的默认格式进行输出，默认的区域是：10-6-30 下午 5:19
按照指定格式 yyyy-MM-dd hh:mm:ss EE，系统默认区域是：2010-06-30 05:19:00 星期三
按照日期的 FULL 模式，区域设置为中文：2010 年 6 月 30 日 星期三
按照指定格式 yyyy 年 MM 月 dd 日 hh 时 mm 分 ss 秒 EE，区域为中文：2010 年 06 月 30 日 05 时 19 分
00 秒 星期三
按照指定格式 yyyy-MM-dd hh:mm:ss EEEEEE，区域是美国：2010-06-30 05:19:00 Wednesday
```

【代码解析】以上程序中，首先创建 Data 对象和 DataFormat 对象，并设置其时间格式（第 05～15 行）。然后按照所设置的格式对时间进行打印输出（第 17～26 行）。

19.4 习题

【习题 19-1】输出 Integer、Float 和 Double 类的最大/最小值，并输出其十六进制形式。

【分析】该题主要考查 Integer、Float 和 Double 类。

【核心代码】

```
public class NumberOne {
   public static void main(String[] args) {
      Integer IntegerMax = Integer.MAX_VALUE;
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
        Integer IntegerMin = Integer.MIN_VALUE;
        System.out.println("Integer 的最大值"+ IntegerMax);
        System.out.println("Integer 的最大值(16 进制)"+
                           Integer.toHexString(IntegerMax));
        System.out.println("Integer 的最大值(8 进制)"+
                           Integer.toOctalString(IntegerMax));
        System.out.println("Integer 的最大值(2 进制)"+
                           Integer.toBinaryString(IntegerMax));
        System.out.println("Integer 的最小值"+ IntegerMin);
        System.out.println("Integer 的最小值(16 进制)"+
                           Integer.toHexString(IntegerMin));
        System.out.println("Integer 的最小值(8 进制)"+
                           Integer.toOctalString(IntegerMin));
        System.out.println("Integer 的最小值(2 进制)"+
                           Integer.toBinaryString(IntegerMin));

        Float floatMax = Float.MAX_VALUE;
        Float floatMin = Float.MIN_VALUE;
        System.out.println("Float 的最大值"+ floatMax);
        System.out.println("Float 的最大值(16 进制)"+ Float.toHexString(floatMax));
        System.out.println("Float 的最小值"+ floatMin);
        System.out.println("Float 的最小值(16 进制)"+ Float.toHexString(floatMin));

        Double doubleMax = Double.MAX_VALUE;
        Double doubleMin = Double.MIN_VALUE;
        System.out.println("Double 的最大值: "+ doubleMax);
        System.out.println("Double 的最大值(16 进制): "+
                           Double.toHexString(doubleMax));
        System.out.println("Double 的最小值: "+ doubleMin);
        System.out.println("Double 的最小值(16 进制): "+
                           Double.toHexString(doubleMin));
    }
}
```

【习题 19-2】有一个字符串形式的数字，分别转换为 Integer、Float 和 Double 类型，然后转换成 int 类型。

【分析】该题主要考查 Integer、Float 和 Double 类。

【核心代码】

```
public class NumberTwo {
   public static void main(String[] args) {
      String s = "3456";
      Integer i = Integer.valueOf(s);
      Float f = Float.valueOf(s);
      Double d = Double.valueOf(s);

      System.out.println("Integer:" + i);
      System.out.println("Float:" + f);
      System.out.println("Double:" + d);

      System.out.println("从 Integer 得到 int: " + i.intValue());
      System.out.println("从 Float 得到 int: " + f.intValue());
      System.out.println("从 Double 得到 int: " + d.intValue());
   }
}
```

【习题 19-3】分别声明 Integer、Float 和 Double 类型的数字，然后分别输出其他两个类型的数字。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

【分析】该题主要考查 Integer、Float 和 Double 类。

【核心代码】

```
public class NumberThree {
   public static void main(String[] args) {
      Integer i = 123;
      Float f = 456f;
      Double d = 789d;

      System.out.println("将 Integer "+ i + " 转换成 Float: "+ i.floatValue());
      System.out.println("将 Integer "+ i + " 转换成 Double: "+ i.doubleValue());

      System.out.println("将 Float "+ f + " 转换成 Integer: "+ f.intValue());
      System.out.println("将 Float "+ f + " 转换成 Double: "+ f.doubleValue());

      System.out.println("将 Double "+ d + " 转换成 Integer: "+ d.intValue());
      System.out.println("将 Double "+ d + " 转换成 Float: "+ d.floatValue());
   }
}
```

【习题 19-4】分别声明 Boolean 和 Byte 类型的变量，然后分别输出它们的值。

【分析】该题主要考查 Boolean 和 Byte 类。

【核心代码】

```
public class BooleanByte {
   public static void main(String[] args) {
      Boolean b = new Boolean(true);
      Byte bt = new Byte((byte) 333);
      System.out.println(b);
      System.out.println(bt);
   }
}
```

【习题 19-5】使用 StringBuffer 类连接字符串，并计算新字符串的长度。

【分析】该题主要考查 StringBuffer 类。

【核心代码】

```
public class StringBufferOne {
   public static void main(String[] args) {
      StringBuffer sb = new StringBuffer();
      sb.append("荷叶生时春恨生，");
      sb.append("荷叶枯时秋恨成。");
      sb.append("深知身在情常在，");
      sb.append("怅望江头江水声。");
      System.out.println(sb);
      System.out.println(sb.length());
   }
}
```

【习题 19-6】使用 StringBuffer 类连接字符串，然后输出指定位置的字符。

【分析】该题主要考查 StringBuffer 类。

【核心代码】

```
public class StringBufferTwo {
   public static void main(String[] args) {
      StringBuffer sb = new StringBuffer();
      sb.append("荷叶生时春恨生，");
      sb.append("荷叶枯时秋恨成。");
      sb.append("深知身在情常在，");
      sb.append("怅望江头江水声。");
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



```
        for (int i = 0; i < sb.length(); i++) {
            System.out.println(sb.charAt(i));
        }
    }
}
```

【习题 19-7】使用 StringBuffer 类连接字符串，然后删除指定位置的字符。

【分析】该题主要考查 StringBuffer 类。

【核心代码】

```
public class StringBufferThree {
    public static void main(String[] args) {
        StringBuffer sb = new StringBuffer();
        sb.append("荷叶生时春恨生，");
        sb.append("荷叶枯时秋恨成。");
        sb.append("深知身在情常在，");
        sb.append("怅望江头江水声。");
        for (int i = 0; i < sb.length(); i++) {
            if ('恨' == sb.charAt(i) || '长' == sb.charAt(i) || '水' == sb.charAt(i)) {
                sb.deleteCharAt(i);
            }
        }
        System.out.println(sb);
    }
}
```

【习题 19-8】使用 StringBuffer 类连接字符串，替换指定位置的字符后输出。

【分析】该题主要考查 StringBuffer 类的综合使用。

【核心代码】

```
public static void main(String[] args) {
    StringBuffer sb = new StringBuffer();
    sb.append("荷叶生时春恨生，");
    sb.append("荷叶枯时秋恨成。");
    sb.append("深知身在情常在，");
    sb.append("怅望江头江水声。");
    for (int i = 0; i < sb.length(); i++) {
        if ('恨' == sb.charAt(i)) {
            sb.replace(i, i+1, "已");
        }
        if ('长' == sb.charAt(i)) {
            sb.replace(i, i+1, "永");
        }
        if ('水' == sb.charAt(i)) {
            sb.replace(i, i+1, "流");
        }
    }
    System.out.println(sb);
}
```

【习题 19-9】输出当前日期，并用毫秒表示自 1970 年 1 月 1 日 00:00:00 以后的时间。

【分析】该题主要考查 Date 类。

【核心代码】

```
public class PrintDate {
    public static void main(String[] args) {
        Date d = new Date();
        System.out.println("当前时间: "+ d);
        System.out.println("当前时间: "+ d.getTime());
    }
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
}
```

【习题 19-10】分别用几种不同的格式输出当前日期。

【分析】该题主要考查 DateFormat 类。

【核心代码】

```
public class FormatDate {
   public static void main(String[] args) {
      Date d = new Date();
      DateFormat df1 = DateFormat.getInstance();
      DateFormat df2 = DateFormat.getDateInstance(DateFormat.FULL, Locale. CHIN-
                                                  ESE);
      DateFormat df3 = new SimpleDateFormat("yyyy-MM-dd HH:MM:SS");
      DateFormat df4 = new SimpleDateFormat("yyyy 年 MM 月 dd 日 HH:MM:SS");
      DateFormat df5 = new SimpleDateFormat("HH:MM:SS");
      System.out.println("当前时间: "+ df1.format(d));
      System.out.println("当前时间: "+ df2.format(d));
      System.out.println("当前时间: "+ df3.format(d));
      System.out.println("当前时间: "+ df4.format(d));
      System.out.println("当前时间: "+ df5.format(d));
   }
}
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第 20 章　容　　器

顾名思义，容器就是用于存放物体的器皿。最经典的容器示例是动漫机器猫的大肚子，它几乎可以装下所有的东西，容器就有这样的功能，它不像数组的空间，数组在初始化的时候就已经将空间确定，而容器在初始化的时候不指定最大的存放空间，系统会根据存放的对象来跳转内存空间。本章将主要介绍容器的使用。



20.1　容器概述

容器在各种类型的应用中使用比较广泛，例如，最常用到的数据库，从数据库中读取出数据之后，通常会先存放到容器中，而且还可能会根据一定的顺序进行排列，就像把一堆书放进机器猫的肚子里，然后告诉机器猫，分类整理好后再拿出来。

容器的种类比较多，大体上可以分为 Collection 和 Map 两种类型。Collection 的意思就是集合，指按照一定的规则存放在一起的对象集，其中包括 List 和 Set，存放在 List 中的对象要按照一定的顺序排放；Set 中无法存放重复的元素。

Map 是指一种对应关系，就像一个学号对应一个学生、一个班级对应一群人一样。这就体现了 Java 的对象引用设计思想，将对象和一个简单的名字进行关联，用户就可以直接根据这个名字来取用对象。



20.2　迭代器

迭代器主要用来对容器中的元素进行遍历，也就是将机器猫肚子里的所有东西都拿出来，并可以对这些东西进行删除等操作。每一个迭代器对象都代表着容器中的一个确定的地址。迭代器有一些比较常用的方法，具体如下。

- boolean hasNext()：该方法的作用是判断是否存在可以访问的元素。
- Object next()：该方法的作用是返回下一个元素。如果程序运行到集合的末尾，则会抛出 NoSuchElementException 异常。
- void remove()：该方法的作用是删除上次访问的元素。

【范例 20-1】示例代码 20-1 是一个使用拆分示例。

示例代码 20-1

```
01  public class CollectionTest {
02          public static void main(String[] args) {
03              Collection<String> collstr = new ArrayList<String>();
04              //创建一个元素为 String 类型的列表
05              collstr.add("张三");                    //为列表添加元素
06              collstr.add("李四");
07              collstr.add("王五");
08              collstr.add("赵六");
09              System.out.println(collstr);            //输出类表的元素内容
10              Object[] arrayo = collstr.toArray();    //将列表转换为数组
```

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
11              Collection<String> coll = new ArrayList<String>();
12              //创建一个元素为字符串类型的列表
13              coll.add("赵刚");              //为列表添加元素
14              coll.add("莉莉");
15              coll.addAll(collstr);
16              System.out.println(coll); //输出第二个集合中的元素内容
17              System.out.println("coll.contains(\"赵刚\") = "+coll.contains("
                                                                  赵刚"));
18              //查看列表是否包含“赵刚”
19              System.out.println("coll.contains(collstr) =
                                                  "+coll.containsAll(collstr));
20              //判断第二个集合是否包含有第一个集合中的元素
21              coll.retainAll(collstr);//从第二个集合中删除第一个集合中不包含的元素
22              System.out.println("从第二个集合中删除第一个集合中不包含的元素，第二个集
                                    合元素为");
23              //输出字符串信息
24              System.out.println(coll);                    //输出第二个集合中的元素
25              coll.add("赵刚");                            //为列表 c2 添加元素
26              Iterator it = coll.iterator();               //返回列表迭代器
27              System.out.print("[");                       //输出字符串信息
28              while(it.hasNext())                          //通过迭代器遍历列表元素
29                  System.out.print(it.next()+", ");        //输出容器元素
30              System.out.println(")");                     //输出字符串信息
31              coll.removeAll(collstr);        //从 coll1 列表中删除 collstr 的元素
32              System.out.println("从第二个集合中删除第一个集合中包含的元素，第二个集合
                                    元素为");
33              //输出字符串信息
34              System.out.println(coll);                    //输出第二个集合中的元素
35              coll.remove("赵刚");                         //删除元素“赵刚”
36          System.out.println("coll.isEmpty() = "+coll.isEmpty());
                                                             //查看类表是否为空列表
37              System.out.println(collstr);                 //输出第一个集合
38              collstr.clear();//清除第一个集合中的元素
39              System.out.println("使用方法 clear()后，第一个集合中的元素
                                    \n"+collstr);
40          }
41      }
```

程序运行结果如图 20-1 所示。

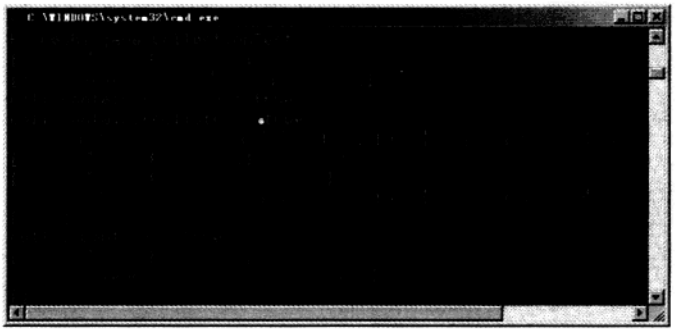


图 20-1　CollectionTest.java 运行结果

【代码解析】在以上程序中，首先创建对象是使用 ArrayList 创建的 collatr 容器对象（第 03 行），添加元素使用 add()方法（第 05～08 行）。接下来创建第二个容器对象 coll（第 11 行），

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

使用 add()方法添加元素（第 13、14 行）。然后使用 coll 的 addAll()方法将第一个集合中的元素添加到第二个集合中（第 15 行）。最后使用 retainAll()方法将第一个集合中的元素剔除（第 31 行），并且使用 clear 方法将第一个集合中的元素全部清除（第 38 行）。

20.3　List

List 接口继承了 Collection 接口，是一个用于有序存放对象的集合，List 中主要包括 ArrayList 和 LinkedList 两个子类。本节将对 List 链表进行详细的介绍。

20.3.1　List 接口方法

List 接口中提供了一些常用的方法，用于对 List 的元素进行操作，具体方法如下。

- void add(int index, Object element)：作用是在指定位置添加元素。
- boolean addAll(int index, Collection c)：作用是在指定位置添加集合。
- ListIterator listIterator()：作用是返回类别列表迭代器。
- List subList(int fromIndex, int toIndex)：作用是返回指定的开始位置(包含)和结束位置(不包含)的子列表。
- ListIterator listIterator(int index) ：作用是返回一个列表迭代器，并且在指定位置开始访问元素。
- Object get(int index) ：作用是返回指定位置元素。
- Object set(int index, Object element) ：作用是设置指定位置的元素，并返回旧的元素。
- int indexOf(Object o) ：作用是返回元素 o 第一次出现的位置，集合不包含该对象时，返回–1。
- int lastIndexOf(Object o) ：作用是返回元素最后出现的位置，集合不包含该对象时，返回–1。
- Object remove(int index) ：作用是删除指定位置的元素，返回被删除的元素。



注意：对 subList()方法返回的子列表的更改（如 add()、remove()和 set()调用）将会影响底层 List。

20.3.2　链表迭代器接口方法

链表的迭代器接口是 ListIteator，该接口继承了 Iteator 接口，并且对其进行扩展。链表迭代器主要支持对 List 的访问，ListIterator 提供的新方法如下。

- void add(Object o) ：作用是将对象 o 添加到集合，位置位于当前对象前。
- void set(Object o) ：作用是设置 next 或 previous 方法访问上一个元素为 o，但是如果上次访问后，列表结构被修改，抛出 IllegalStateException 异常。
- boolean hasPrevious()：作用是回向后迭代时是否有元素可以访问。
- Object previous()：作用是返回列表的前一个元素。
- int nextIndex()：作用是返回调用 next 方法所返回元素的索引。
- int previousIndex()：作用是返回调用 previous 方法所返回元素的索引。

以上方法中，如果采用 add()方法添加元素，则元素会存放在前面，所以添加元素之后，再调用 previous()才可以返回新元素。而 ListIterator 中的 remove()和 set()方法并不是针对当前游

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

标的操作，而是对最后一次调用 next()或者 previous()的操作。

20.3.3 链表和数组链表

Java 中的集合由 LinkList 和 ArrayList 两个实现类组成，后者支持随机访问，而且不需要在末尾以外的地方插入或者删除集合。而 LinkedList 可以比较频繁地对列表中间的元素进行删除或者添加操作。用户可以根据使用时的不同需求来选择不同的实现类。

LinkedList 中实现了 List 接口，并且允许 null 元素，而且在 LinkedList 的首部或尾部还提供有额外的 get()、remove()和 insert()方法。这些操作使 LinkedList 可被用做堆栈（stack）、队列（queue）或双向队列（deque）。

```
LinkedeList list=new LikedeList();//创建空链表
```



注意：在 LinkedList 中不存在同步方法。所以出现多个线程同时访问一个 List 时，则必须自己实现访问同步，其中一个解决方法是在创建 List 的同时再构造一个用于同步的 List。

```
List list = Collections.synchronizedList(new LinkedList(...));
```

在 ArrayList 和 LinkedList 中各自都提供了两个构造方法，分别是无参构造方法和 Collection 集合的构造方法，并且二者都实现了 Cloneable 接口。LinkedList 类中添加了一些处理列表两端元素的方法，主要方法和功能如表 20-1 所示。

表 20-1 LinkedList 类处理列表两端元素的方法

方 法	功 能
void addFirst(Object o)	将对象 o 添加到链表的头部
void addLast(Object o)	将对象 o 添加到链表的结尾
Object getFirst()	返回链表的头部元素
Object getLast()	返回链表末尾的元素
Object removeFirst()	删除链表的头部元素，并返回
Object removeLast()	删除链表的末尾元素，并返回
LinkedList()	构建一个空的链接列表
LinkedList(Collection c)	构建一个链接列表，将集合 c 的元素添加到其中

20.3.4 List 应用

1. 链表应用

对于链表的应用，首先是在 MyLinkedList 类中创建内部类 AbsMyList 以及接口 ImpQueue 和 ImpStack 接口。

【范例 20-2】示例代码 20-2 是一个使用拆分示例。

示例代码 20-2

```
01  abstract class AbsMyList {
02      protected int len = 0;          // 长度
03      protected AbsMyList() {         // 默认构造方法
04      }
05      abstract public boolean add(Object o);          // 向链表末尾添加一个元素
06      abstract public Object getElement(int index);   // 获得指定元素
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
07      abstract public boolean isElement(Object o);     // 判断是否包含元素
08      abstract int indexOf(Object o);                  // 判断元素的位置
09      abstract public boolean deleteElement(Object o);// 移除元素
10      abstract public void clear();                    // 清空
11      public int len() {                               // 获得长度大小
12          return this.len;
13      }
14      public boolean isEmpty() {                       // 判断是否为空
15          return this.len == 0;
16      }
17  }
18  interface ImpQueue {                                 // 队列接口
19      void inQueue(String o);                          // 入队
20      Object outQueue();                               // 出队
21      boolean isEmpty();                               // 判断是否为空
22  }
23  interface ImpStack {                                 // 栈接口
24      void push(Object o);                             // 元素入栈
25      Object pop();                                    // 元素出栈
26      boolean isEmpty();                               // 是否为空
27  }
```

【代码解析】在以上程序中，首先创建内部类 AbsMyList（第 01～17 行），然后创建接口 ImpQueue 和 ImpStack，并在接口中定义一系列的方法（第 18～27 行）。

MyLinkedList 中创建一个内部类 LinkedList，继承 AbsMyList 并实现 ImpQueue 和 ImpStack 接口。

MyLinkedList 类

```
01  class LinkedList extends AbsMyList implements ImpQueue, ImpStack {
02      private static class InnerMyEntry {              // 静态内部类
03          Object object;                               // 对象
04          InnerMyEntry next;                           // 下一个对象
05          InnerMyEntry previous;                       // 上一个对象
06          // 对带参数的构造方法进行初始化
07          InnerMyEntry(Object object, InnerMyEntry next, InnerMyEntry previous) {
08              this.object = object;
09              this.next = next;
10              this.previous = previous;
11          }
12      }
13      private InnerMyEntry header = new InnerMyEntry(null, null, null);
                                                         // 创建实体对象
14      public LinkedList() {                            // 默认构造方法进行初始化
15          this.header.next = this.header.previous = this.header;
16      }
17  private InnerMyEntry addBefore(Object o, InnerMyEntry e) {
                                                         // 添加对象之前的操作方法
18  InnerMyEntry newEntry = new InnerMyEntry(o, e, e.previous);  // 创建实体对象
19          newEntry.previous.next = newEntry;
20          newEntry.next.previous = newEntry;
21          this.len++;
22          return newEntry;
23      }
24      public boolean add(Object o) {                   // 添加对象
25          this.addBefore(o, header);
26          return true;
27      }
28      public void clear() {                            // 清空对象
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
29          InnerMyEntry e = this.header.next;
30          while (e != this.header) {          // 判断进行循环
31              InnerMyEntry next = e.next;
32              e.next = e.previous = null;     // 清空对象
33              e.object = null;                // 清空对象
34              e = next;
35          }
36          this.header.next = this.header.previous = this.header;
37          this.len = 0;
38      }
39      public boolean isElement(Object o) {    // 是否包含对象
40          return this.indexOf(o) != -1;
41      }
42      public Object getElement(int index) {   // 获得指定的对象
43          InnerMyEntry myEntry = this.entry(index);
44          if (myEntry == null)
45              return null;
46          return myEntry.object;
47      }
48      public int indexOf(Object o) {          // 获得对象在栈或队列中的位置
49          int index = 0;
50          if (o == null) {
51              for(InnerMyEntry e=this.header.next;e!=this.header;e=e.next)
    {
52      // 循环获得元素对象
53                  if (e.object == null)
54                      return index;
55                  index++;
56              }
57          } else {
58              for(InnerMyEntry e=this.header.next;e!=this.header;e=e.next)
    {
59      // 循环获得元素对象
60                  if (o.equals(e.object))
61                      return index;
62                  index++;
63              }
64          }
65          return -1;
66      }
67      public boolean deleteElement(Object o) {        // 移除对象
68          if (o == null) {
69                                                      // 循环获得元素对象
70              for (InnerMyEntry e = header.next; e != header; e = e.next) {
71                  if (e.object == null) {
72                      return this.deleteElement(e); // 移除对象
73                  }
74              }
75          } else {
76          for (InnerMyEntry e = header.next; e != header; e = e.next) {
                                                      // 循环获得元素对象
77                  if (o.equals(e.object)) {
78                      return this.deleteElement(e); // 移除对象
79                  }
80              }
81          }
82          return false;
83      }
84      public Object outQueue() {                      // 出队方法
85          Object result = this.header.next.object;    // 获得对象
86          this.deleteElement(this.header.next);       // 移除对象
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
87          return result;
88      }
89      public void inQueue(String o) {                    // 入队方法
90          this.addBefore(o, header);                     // 调方法添加对象
91      }
92      public Object pop() {                              // 出栈方法
93          Object result = this.header.previous.object;// 获得对象
94          this.deleteElement(this.header.previous);      // 移除对象
95          return result;
96      }
97      public void push(Object o) {                       // 入栈
98          this.addBefore(o, header);                     // 调方法添加对象
99      }
100     private boolean deleteElement(InnerMyEntry e) { // 移除对象
101         if (e == header) {
102             return false;
103         }
104         e.previous.next = e.next;                      // 重新赋值
105         e.next.previous = e.previous;                  // 重新赋值
106         e.next = e.previous = null;                    // 清空
107         e.object = null;
108         this.len--;
109         return true;
110     }
111     private InnerMyEntry entry(int index) {            // 获得指定的对象
112         if (index < 0 || index >= this.len) {          // 判断指定元素的下标
113             return null;
114         }
115         InnerMyEntry e = header;
116         if (index < (this.len >> 1)) {                 // 判断循环获得指定的实体
117             for (int i = 0; i <= index; i++)
118                 e = e.next;
119         } else {
120             for (int i = this.len; i > index; i--)
121                 e = e.previous;
122         }
123         return e;
124     }
125 }
```

【代码解析】以上程序中，首先创建接口类和接口的实现类 LinkedList，然后实现 ImpQueue 和 ImpStack 两个接口（第 01 行）。接着在此类中创建一个内部类 InnerMyEntry 类（第 02 行），并在该内部类中声明两个自身类的对象，然后利用带参数的构造方法为声明的对象赋值（第 03～15 行）。使用 add()方法调用 addBefore()方法并返回真值（第 24～27 行）。clear()方法获得下一个对象，根据对象不是当前对象进行循环，获得这个对象的下一个对象，并将这个对象的下一个与上一个对象置空（第 28～38 行）。isElement()方法调用 indexOf()方法判断对象是否存在，如果不存在，则返回 false（第 39～41 行）。getElement()方法根据编号获得指定的对象，如果对象不为空，则返回对象的元素（第 42～47 行）。outQueue()方法获得对象的元素，调用 deleteElement()方法移除该对象，并返回该对象（第 84～88 行）。inQueue()方法调用 addBefore()方法往对象中添加元素（第 89～91 行）。pop()方法获得对象，调用 deleteElement()方法移除该对象，并返回该对象（第 92～96 行）。

最后要在类 MyLinkedList 中创建主方法，并调用以上内部类中的方法打印输出。

MyLinkedList

```
01  public class MyLinkedList {
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
02      public static void main(String[] args) {
03          ImpQueue myQueue = new LinkedList();    // 实例化队列 Queue 对象
04          myQueue.inQueue("\tThankyou");          // 添加元素入队
05          myQueue.inQueue("\tThanks");            // 添加元素入队
06          System.out.println("Queue 队列中元素出队情况：");
07          while (!myQueue.isEmpty()) {            // 循环判断队列是否为空
08          System.out.println(myQueue.outQueue());// 不为空则输出队列中的元素
09          }
10          ImpStack myStack = new LinkedList();    // 实例化 Stack 栈对象
11          System.out.println("\nStack 栈中元素出栈情况：");
12          myStack.push("\tabcdefg");              // 添加元素入栈
13          myStack.push("\tgfedcba");              // 添加元素入栈
14          while (!myStack.isEmpty()) {            // 循环判断栈是否为空
15              System.out.println(myStack.pop());// 不为空则输出出栈元素
16          }
17      }
18  }
```



图 20-2　MyLinkedList.java 运行结果

【代码解析】以上程序中，首先创建主方法，并在主方法中分别调用内部类中的各种方法，传入参数（第 02～13 行），然后打印输出出栈元素（第 15 行）。程序运行结果如图 20-2 所示。

2. 数组链表应用

下面是一个 ArrayList 的例子，该例为数组链表添加多个元素，并判断该数组链表是否包含某个元素，使用了 for 循环对数组链表元素遍历。

【范例 20-3】示例代码 20-3 是一个使用拆分示例。

示例代码 20-3

```
01  public class ArrayListTest {
02      public static void main(String[] args) {
03          List<String> list = new ArrayList<String>(4); //创建一个大小为 4 的集合
04          list.add("语文");              // 向集合添加元素，增加科目
05          list.add("数学");
06          list.add("生物");
07          list.add("物理");
08          System.out.println("第一次选课单中的科目：");
09          for (Iterator<String> iter = list.iterator(); iter.hasNext();) {
10              String s = iter.next();   // 获取每个元素
11              System.out.println(s + "\t");
12              if ("数学".equals(s)) {    // 第一次选课的科目为数学
13                  iter.remove();         // 将其从科目列表中删除
14              }
15          }
16          System.out.println();
17          System.out.println("第二次选课单中的科目：");
18          Iterator<String> it = list.iterator();
19          while (it.hasNext()) {
20              String s = it.next();
21              System.out.println(s + "\t");
22              if ("语文".equals(s) || "物理".equals(s)) {// 第二次选择语文和物理
23                  it.remove();
24              }
25          }
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
26          System.out.println();
27          System.out.println("最后还没有选择的科目：");
28          for (Iterator<String> iter = list.iterator(); iter.hasNext();) {
                                                        //  打印剩余科目
29              String s = iter.next();
30              System.out.println(s + "\t");
31          }
32      }
33  }
```

【允许结果】程序运行结果如图 20-3 所示。

【代码解析】以上程序中，首先创建 ArrayList 动态数组对象 list（第 03 行），使用 add 方法在数组中添加元素（第 04～07 行）。然后使用循环一次次过滤掉已经选择了的科目（第 12～33 行），最后剩下生物，将其打印输出。

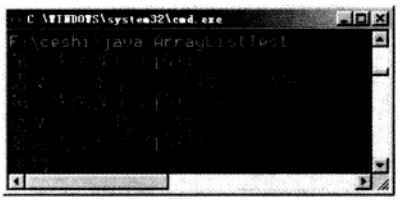


图 20-3　ArrayListTest.java 运行结果

20.4　集 Set

Set 接口具有和 Collection 完全一样的接口。Set 和 List 最大的不同就是 Set 中的成员不能够重复，也就是说，Set 中的任意两个成员之间相互调用方法 equals 必须返回为 false。equals 方法保证元素的唯一性。

20.4.1　Set 接口的方法

常用的 Set 有以下三个，它们都实现了 Set 接口，并且具备各自的特点。

- HashSet：为快速查找而设计的 Set，存入 HashSet 的对象必须定义 HashCode 方法。
- TreeSet：为了确保 Set 中成员的有序而设计的一个 Set 的子类，存入 TreeSet 中的对象如果实现了 Comparable 接口，则要按照接口的 CompareTo 方法比较大小后按照升序排列，程序员也可以在构造方法中直接指定比较器。
- LikedHashSet：具有 HashSet 的查询速度，而且底层使用的链表维护元素的顺序，也就是元素的顺序与插入的顺序相同。

SortedSet 是 Set 的一个子类，除了具有 Set 不能保存相等的对象这一特点外，存入其内部的元素是有序的。SortedSet 接口为集合的两端提供了访问方法，当处理类别的子集时，会影响到源集合。SortedSet 提供的方法如下。

- Comparator comparator()：该方法的作用是返回对集合内的元素进行排序的比较器，如果实现了 Comparable 接口的 compareTo()，则返回 null。
- Object first()：该方法的作用是返回集合第一个元素。
- Object last()：该方法的作用是返回集合中最后一个元素。
- SortedSet subSet(Object fromElement, Object toElement)：该方法的作用是返回从 fromElement(包括)至 toElement(不包括)范围的子集。
- SortedSet headSet(Object toElement)：该方法的作用是返回子集，子集元素小于 toElement。
- SortedSet tailSet(Object fromElement)：该方法的作用是返回子集，子集元素大于或等于 toElement。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

20.4.2 哈希集和树集

Set 接口中有两个实现类，分别是 HashSet 和 TreeSet。在需要自己手动创建类时，需要覆盖 hashCode()方法和 equals()方法。其中 TreeSet 是以有序的方式对元素进行排列。HahsSet 和 TreeSet 都有自己不同的构造方法。

1. 哈希集 HashSet 类构造方法

- HashSet()：该方法的作用是创建一个空的哈希集合。
- HashSet(Collection c)：该方法的作用是创建一个哈希集合，并将 c 集合中的元素添加至其中。
- HashSet(int initialCapacity)：该方法的作用是创建具有一定容量的哈希集合。
- HashSet(int initialCapacity, float loadFactor)：该方法的作用是创建具有一定容量和加载因子(0.0～10)的空哈希集。

2. 树集 TreeSet 类构造方法

- TreeSet()：创建空的树集合。
- TreeSet(Collection c)：该方法的作用是创建一个树集合，并添加集合 c 的元素。
- TreeSet(Comparator c)：该方法的作用是创建树集合，并制定特定的比较器进行排序。

LinkedHashSet 是对 HashSet 的扩展，而且它具有 HashSet 的查询速度，且底层使用链表维护元素的顺序，也就是说，元素的顺序与所插入的顺序相同。

【范例 20-4】示例代码 20-4 是一个 LinkHashSet 示例。

示例代码 20-4

```
01  public class SetTest {
02      public static void main(String[] args) { // java 程序主入口处
03          hashSetResult();
04          treeSetResult();
05          linkedHashSetResult();
06          Set hashSet = new HashSet();
07          addSetElement(hashSet);
08          hashSet.add("数学");                    // Set 不允许元素重复
09          hashSet.add("语文");
10          System.out.println("\n 为 hashSet 加入数学，语文元素后->: ");
11          showElement(hashSet);                   // 调用方法显示元素
12          hashSet.remove("物理");                 // 删除元素
13          System.out.println("\nhashSet 删除物理元素后->: ");
14          showElement(hashSet);                   // 调用方法显示元素
15          List list = new ArrayList();            // 创建一个列表集合
16          list.add("数学");
17          list.add("化学");
18          list.add("生物");
19          hashSet.addAll(list);                   // 将列表集合添加到 Set 中
20          System.out.println("\nhashSet 添加一个集合的所有元素后->: ");
21          showElement(hashSet);
22          hashSet.retainAll(list);                // 删除除列表集合中的元素之外的元素
23          System.out.println("\nhashSet 删除除列表集合之外的元素后->: ");
24          showElement(hashSet);                   // 调用方法显示元素
25          hashSet.removeAll(list);                // 删除集合中的元素
26          System.out.println("\nhashSet 删除集合中的元素后->: ");
27          showElement(hashSet);                   // 调用方法显示元素
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
28          // 获取 Set 中元素的个数
29          System.out.println("\thashSet 中当前元素的个数: " + hashSet.size());
30          // 判断 Set 中的元素是否为空
31          System.out.println("\thashSet 中当前元素为 0?  " + hashSet.isEmpty());
32      }
33      public static void hashSetResult() {    // 使用 HashSet 操作元素
34          Set hashSet = new HashSet();
35          addSetElement(hashSet);             // 调用方法初始化元素
36          System.out.println("使用 HashSet 存储的数据元素如下->: ");
37          showElement(hashSet);               // 调用方法显示元素
38      }
39      public static void treeSetResult() {    // 使用 TreeSet 操作元素
40          Set treeSet = new TreeSet();
41          addSetElement(treeSet);             // 调用方法初始化元素
42          System.out.println("\n 使用 TreeSet 存储的数据元素如下->: ");
43          showElement(treeSet);               // 调用方法显示元素
44      }
45      public static void linkedHashSetResult() { // 使用 LinkedHashSet 操作元素
46          Set linkedHashSet = new LinkedHashSet();
47          addSetElement(linkedHashSet);       // 调用方法初始化元素
48          System.out.println("\n 使用 LinkedHashSet 存储的数据元素如下->: ");
49          showElement(linkedHashSet);         // 调用方法显示元素
50      }
51      public static void addSetElement(Set set) { // 初始化 Set 的元素
52          if (set != null) {
53              set.add("数学");
54              set.add("语文");
55              set.add("生物");
56              set.add("化学");
57              set.add("政治");
58          }
59      }
60      public static void showElement(Set set) {   // 输出 set 的元素
61          if (set != null && set.size() > 0) {
62              Iterator it = set.iterator();       // 获得迭代器 Iterator
63              while (it.hasNext()) {              // 循环获得 Set 中的每个元素
64                  System.out.print(it.next() + " ");
65              }
66          } else {
67              System.out.println("没有元素! ");
68          }
69          System.out.println();                   // 换行
70      }
71  }
```

【运行结果】程序运行结果如图 20-4 所示。

【代码解析】以上程序中，共定义了六个方法，它们分别是 addSetElement()方法、showElement()方法、hashSetResult()方法、treeSetResult()方法、linkedHashSetResult()方法以及 main()方法（第 33、第 39、第 45、第 51、第 60 和第 02 行），分别用来初始化 set 以及向集合中添加元素。然后分别打印输出。最后执行删除集合之外的元素，并打印输出。

20.4.3　实现 Set

下面是一个演示 TreeSet 的例子，该程序中元素的类型为 Student 类，Student 类实现了 Comparable 接口，用于对元素进行排序。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

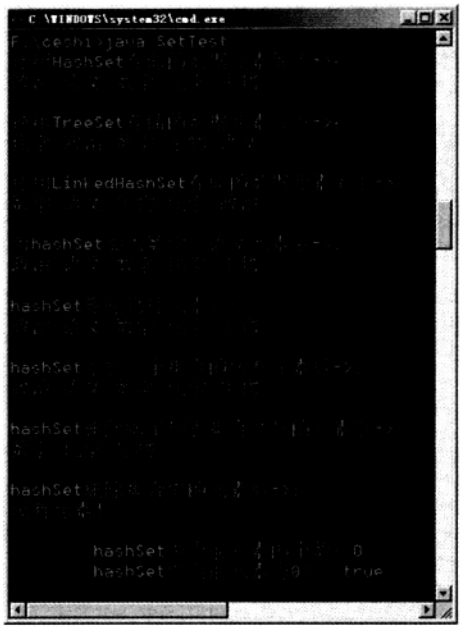


图 20-4　SetTest.java 运行结果

【范例 20-5】示例代码 20-5 是一个使用拆分示例。

示例代码 20-5

```
01  public class TreeSetTest {
02            public static void main(String[] args) {
03              SortedSet<People> people=new TreeSet<People>();
                  //创建一个排序的集合
04            People people1=new People("张三","0001");
                  //创建并初始化 Student 对象
05              People people2=new People("李四","0002");
06                    People people3=new People("王五","0003");
07                    People people4=new People("赵六","0004");
08                    People people5=new People("钱七","0005");
09                    People people6=new People("赵刚","0006");
10                    people.add(people1);                    //将元素 s1 添加到集合
11                    people.add(people2);
12                    people.add(people3);
13                    people.add(people4);
14                    people.add(people5);
15                    people.add(people6);
16                    System.out.print(people);        //输出集合元素
17              }
18              public static void print(Collection c){
19                    Iterator it=c.iterator();        //返回集合迭代器
20                    while(it.hasNext()){             //遍历迭代器
21                          Object o=it.next();
22                          System.out.println(o);     //输出元素
23                    }
24              }
25        }
26        class People implements Comparable {
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
27            String name;                                    //声明字符串类型域 name
28            String id;                                      //声明字符串类型域 id
29            //声明 Student 对象构造方法
30            public People(String name, String id) {
31                  this.name = name;                         //初始化成员对象 name
32                  this.id = id;                             //初始化成员变量 id
33            }
34            //将对象信息转化为字符串信息
35            public String toString() {
36                  return name+"|"+id;                       //返回信息
37            }
38            //实现 compareTo 接口，比较 id 的大小
39            public int compareTo(Object o){
40               People ts=(People)o;                         //创建 People 局部变量
41               return id.compareTo(ts.id);                  //返回 compareTo 运算结果
42            }
43      }
```

【运行结果】运行结果如图 20-5 所示。



图 20-5　TreeSetTest.java 运行结果

【代码解析】以上代码中，通过 TreeSet 创建了一个 SortedSet 对象 people（第 03 行），people 调用 add 方法添加 Student 对象作为 people 的元素（第 04～09 行）。Student 对象实现了 Comparable 接口的 compareTo 方法，用于比较两个 Student 对象的大小(第 39～41 行)。在 people 添加元素的过程中会自动完成排序的过程，所以，添加完元素后的集合已经是排好序的。



20.5　映射 Map

Map 提供了一种键值对应的存储方式，是指对象和另一个对象形成一一对应的关系，从一个对象来获取另一个对象的存在，不过这种映射关系是单向的。对于键对象来说，像 Set 一样，一个 Map 容器中的键对象不允许有重复，这样就保证了键的唯一性。用户可以根据键对象来快速定位值对象。

20.5.1　Map 接口方法

在 Map 映射接口中也提供了一些常用的方法，具体方法如下。

- boolean containsKey(Object key)：该方法的作用是判断映射是否存在 key 关键字。
- boolean containsValue(Object value)：该方法的作用是判断映射中是否存在值 value。
- void clear()：该方法的作用是删除映射的所有元素。
- Object remove(Object key)：该方法的作用是删除与键对应的映射。
- void putAll(Map t)：该方法的作用是给该映射添加 t 映射的所有元素。
- Object get(Object key)：该方法的作用是返回与键对应的值对象，如果没有找到，则返回 null。
- Object put(Object key, Object value)：该方法的作用是将相关的键值对放入映射。
- int size()：该方法的作用是返回映射的数量。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

- Set entrySet()：该方法的作用是返回 Map.Entry 的视图集合。因为键具有唯一性，它可以删除元素，同时会反映到源映射中，但是不能添加任何元素。
- boolean isEmpty()：该方法的作用是判断映射是否为空。
- Set keySet()：该方法的作用是返回映射关键字的视图集合必须用 Set 支持，因为键具有唯一性，它可以删除元素，同时会反映到源映射中，但是不能添加任何元素。
- Collection values()：该方法的作用是返回映射中值的视图集合，返回类型为 Collection。在返回 Collection 中删除元素，同时会反映到源映射中，但是不能添加任何元素。

SortedMap 比 HashMap 多了一项对存入的键值按照键大小的顺序进行排序的功能，插入排序映射中的键必须实现 Copparable 接口，以便按键值进行排序。如果没有实现，则会按照自然的顺序进行排列。SortedMap 只有一个实现类 TreeMap。SortedMap 提供的方法如下。

- Comparator comparator()：该方法的作用是返回对关键字排序的比较器，如果实现了 Comparable 接口的 compareTo()方法，则返回 null。
- Object firstKey()：该方法的作用是返回映射的第一个关键字。
- Object lastKey()：该方法的作用是返回映射的最后一个关键字。
- SortedMap subMap(Object fromKey, Object toKey)：该方法的作用是返回指定范围(fromKey 到 toKey，不包括 toKey)的子集。
- SortedMap headMap(Object toKey)：该方法的作用是返回 SortedMap 的一个视图，其内各元素的 key 皆小于 toKey。
- SortedSet tailMap(Object fromKey)：该方法的作用是返回 SortedMap 的一个视图，其内各元素的 key 皆大于或等于 fromKey。

20.5.2 哈希映射和树映射

Map 的实现有哈希映射 HashMap 和树映射 TreeMap，两者是根据不同的功能进行选择的，例如，HashMap 比较适合于频繁地在 Map 中进行插入、删除元素。而 TreeMap 则比较适合只按照自然的顺序或者自定义遍历。HashMap 添加的键类必须要重写 hashCode()和 equals()方法，而 TreeMap 则必须要实现 Comparable 接口。

1. 哈希映射 HashMap 类的构造方法

- HashMap()：该方法用于创建一个空的哈希映射。
- HashMap(Map m)：该方法用于创建一个空的哈希映射，并将 m 的所有映射添加到其中。
- HashMap(int initialCapacity)：该方法用于创建一个指定容量的哈希映射。
- HashMap(int initialCapacity, float loadFactor)：该方法用于创建一个指定容量的哈希映射，并指定加载因子。



注意：在使用 HashMap 时，为了优化空间资源的使用，可以指定初始容量和负载因子。

2. 树映射 TreeMap 类的构造方法

- TreeMap()：该方法用于构建一个空的映射树。
- TreeMap(Map m)：该方法用于构建一个映射树，并且添加映射 m 中的所有元素。
- TreeMap(Comparator c)：该方法用于构建一个映射树，并且使用特定的比较器对关键字进行排序。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

- TreeMap(SortedMap s)：该方法用于构建一个映射树，添加映射树 s 中的所有映射，并且使用与有序映射 s 相同的比较器排序。



注意：TreeMap 不可以像 HashMap 一样使用调优选项，因为 TreeMap 处于平衡状态。

20.5.3　哈希映射应用

HashMap 中会通过 get()方法来获取 value，然后会通过 put()来插入 value，用于检验对象是否存在 ContainKey()方法。HashMap 是通过 key 来索引其内容的，除了这方面外，和 ArrayList 的操作没有其他差异。下面是一个 HashMap 和 ArrayList 的对比示例。

【范例 20-6】示例代码 20-6 是一个哈希映射拆分示例。

示例代码 20-6

```
01  public class MapTest {
02      public static void main(String[] args) {// java 程序主入口处
03          printHashMap();                     // 操作 HashMap 显示课程科目
04          printTreeMap();                     // 操作 TreeMap 显示课程科目
05          printArrayList();                   // 操作 ArrayList 显示课程科目
06      }
07      public static void printHashMap() {     // 操作 HashMap 的方法
08          Map map = new HashMap();            // hashMap 是无序的
09          map.put("01", "语文");
10          map.put("02", "数学");
11          map.put("03", "物理");
12          map.put("04", "生物");
13          map.put("05", "化学");
14          map.put("06", null);
15          map.put(null, "政治");
16          Iterator iterator = map.keySet().iterator(); // 获得迭代器 iterator
17          System.out.println("<1> 操作 HashMap 显示课程科目");
18          while (iterator.hasNext()) {                // 循环获得每个对象
19              Object key = iterator.next();           // 获得每个键元素
20              System.out.print(map.get(key) + "\t"); // 输出值
21          }
22          System.out.println();
23      }
24
25      public static void printTreeMap() {             // 操作 TreeMap 的方法
26          TreeMap treeMap = new TreeMap();            // HashMap 是无序的
27          treeMap.put("01", "语文");
28          treeMap.put("02", "数学");
29          treeMap.put("03", "物理");
30          treeMap.put("04", "生物");
31          treeMap.put("05", "化学");
32          treeMap.put("06", null);
33          Iterator iterator = treeMap.keySet().iterator();// 获得迭代器 iterator
34          System.out.println("<2> 操作 TreeMap 显示课程科目");
35          while (iterator.hasNext()) {                    // 循环获得每个对象
36              Object key = iterator.next();               // 获得每个键元素
37              System.out.print(treeMap.get(key) + "\t");  // 输出值
38          }
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
39            System.out.println();
40        }
41        public static void printArrayList() {        //操作有序的 ArrayList 列表集合
42            ArrayList arrayList = new ArrayList(); // 创建列表集合对象
43            arrayList.add("01 语文");
44            arrayList.add("03 物理");
45            arrayList.add("06 地理");
46            arrayList.add("04 生物");
47            arrayList.add("05 化学");
48            arrayList.add("02 数学");
49            System.out.println("<3> 排序前的课程种类");
50            for (int i = 0; i < arrayList.size(); i++) {
                                                        // 循环显示列表集合中的元素
51                System.out.print(arrayList.get(i) + "\t");
52            }
53            System.out.println();
54            Collections.sort(arrayList);              // 对列表集合进行排序
55            System.out.println("<4> ArrayList 排序后课程科目");
56            for (int i = 0; i < arrayList.size(); i++) {
                                                        // 循环显示列表集合中的元素
57                System.out.print(arrayList.get(i) + "\t");
58            }
59            System.out.println();
60        }
61    }
```

【运行结果】运行结果如图 20-6 所示。

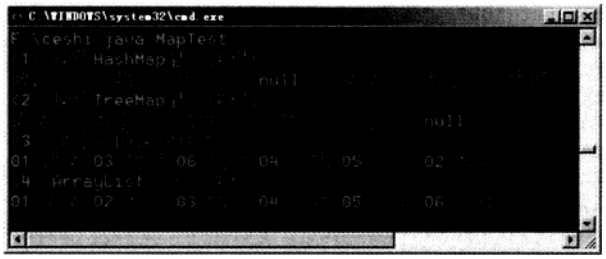


图 20-6　MapTest.java 运行结果

【代码解析】在以上程序中，首先创建操作 HashMap 的方法，并在该方法中为 HashMap 添加元素，使用迭代器打印输出（第 07～23 行）。然后创建一个 TreeMap 方法，并在该方法中添加 TreeMap 元素，使用迭代器打印输出（第 25～40 行）。最后创建 ArrayList 方法，并在方法中添加元素，且打印输出（第 41～60 行）。在主方法中分别调用这三个方法，并对比打印结果（第 02～06 行）。

20.5.4　实现 Map

HashMap 根据键值的 HashCode 值来进行数据的存储，并且根据键可以直接获取它的值，这样，访问的速度就变得非常快。不过 HashMap 也是有缺点的，HashMap 不支持线程的同步，即同一时刻有多个线程同时写 HashMap；HashMap 最多只允许一条记录的键值为 Null，允许多条记录的值为 Null。下面是一个 TreeMap 的例子。

【范例 20-7】示例代码 20-7 是一个使用拆分示例。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

示例代码 20-7

```
01  public class TreeMapTest {
02              public static void main(String[] args) {
03      Map<String, Integer> hm = new HashMap<String, Integer>();
                                                    //创建 HashMap 对象 hm
04            hm.put("张三 ", new Integer(5));      //添加一个映射关系的键值
05            hm.put("李四", new Integer(3));       //添加一个映射关系的键值
06            hm.put("王五 ", new Integer(2));      //添加一个映射关系的键值
07            hm.put("赵六", new Integer(1));       //添加一个映射关系的键值
08            hm.put("钱七", new Integer(4));       //添加一个映射关系的键值
09            hm.put("赵刚", new Integer(7));       //添加一个映射关系的键值
10            hm.put("思思", new Integer(6));       //添加一个映射关系的键值
11            //创建一个 TreeMap 对象
12            Map<String, Integer> sortedMap = new TreeMap<String, Integer>();
13            sortedMap.putAll(hm);                 //将 hm 所有的元素添加到 sortedMap
14            System.out.println(sortedMap);        //输出 sortedMap 的键值对
15            Iterator  itk = hm.keySet().iterator();//返回 hm 的键的迭代器
16            Iterator itv = hm.values().iterator();//返回 hm 值的迭代器
17             while  (itk.hasNext())  {            //通过键的迭代器遍历
18                    Object  key  =  itk.next();   //返回键的下一个元素
19                    Object value = itv.next();    //返回值的下一个元素
20                    System.out.print(key.toString()+"=");        //输出键的名称
21                    System.out.print(value.toString()+", ");     //输出值
22             }
23            }
24            public static void print(Collection c)              //输出集合类
25       {
26                  Iterator it=c.iterator();                     //返回迭代器
27                  while(it.hasNext())                           //遍历集合类
28                  {
29                        Object o=it.next();                     //获取下一个元素
30                        System.out.println(o);                  //输出该元素
31                  }
32            }
33      }
```

【运行结果】程序运行结果如图 20-7 所示。

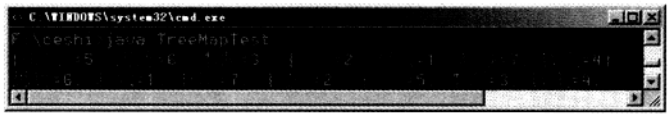


图 20-7　TreeMapTest.java 运行结果

【代码解析】以上程序中，首先创建一个哈希表 hm，并通过 put 方法为哈希表添加一个键值对(第 04～10 行)，然后创建一个 TreeMap 对象 sortedMap，通过 putAll 方法将 hm 的所有元素添加到 sortedMap 中(第 12、13 行)，最后分别提供方法遍历 sortedMap 中的元素(第 17～21 行)。

20.6　习题

【习题 20-1】使用集合编写约瑟夫环程序。

【分析】该约瑟夫环程序也就是幸运儿的问题，内容就是 100 个人围成一圈，开始报数，报到 14 的那个人就要退出圈外，并开始重新计数。依次循环，直到圈中剩下一个人，这个人

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

就是幸运儿。

【核心代码】

```
public static void main(String args[]) {
        // 创建一个元素类型为 Integer 的链表
        LinkedList lucky = new LinkedList();
        int number, cnt;
        // 将每个人的编号依次放入链表中
        for (number = 1; number <= 100; ++number)
            lucky.addLast(number);
        cnt = 100;
        number = 0;
        Iterator it = lucky.iterator();
        // 循环删除退出的人，直到只剩下一个人
        while (cnt > 1) {
            if (it.hasNext()) {
                it.next();              // 往后面数
                ++number;               // 计数器加 1
            } else {                    // 迭代器已经到达末尾，重新将它置回链表头部
                it = lucky.iterator();
            }
            // 删除应该退出圈外的人
            if (number == 14) {
                number = 0;
                it.remove();
                --cnt;
            }
        }
        // 最后链表中剩下的就是幸运儿
        System.out.println("幸运儿编号为: " + lucky.element());
    }
```

【习题 20-2】编写一个查找鱼是谁的宠物程序，该程序主要使用 ArrayList 集合将各种情况打印出来。

【分析】该题的内容如下：在一条街上有五座房子，喷了五种颜色，在每个房里住着不同国籍的人，每个人喝不同的饮料，抽不同品牌的香烟，养不同的宠物，问谁养的是鱼？

【核心代码】

```
public void lookUp() { // 查找符合题意的答案
        init(); // 调用方法实始化数据
        for (int num1 = 0; num1 < co_Array.length; num1++) {
            if (!case4(num1))
                continue;
            if (!case14(num1))
                continue;
            for (int num2 = 0; num2 < pe_Array.length; num2++) {
                if (!case1(num2, num1))
                    continue;
                if (!case8(num2))
                    continue;
                for (int num3 = 0; num3 < dr_Array.length; num3++) {
                    if (!case3(num2, num3))
                        continue;
                    if (!case5(num1, num3))
                        continue;
                    if (!case9(num3))
                        continue;
                    for (int num4 = 0; num4 < sm_Array.length; num4++) {
                        if (!case7(num1, num4))
                            continue;
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
                        if (!case12(num4, num3))
                            continue;
                        if (!case13(num2, num4))
                            continue;
                        if (!case15(num4, num3))
                            continue;
                        for (int num5 = 0; num5 < pet_Array.length; num5++) {
                            if (!case2(num2, num5))
                                continue;
                            if (!case6(num4, num5))
                                continue;
                            if (!case10(num4, num5))
                                continue;
                            if (!case11(num5, num4))
                                continue;
                            total++;
                            answer(num1, num2, num3, num4, num5);
                        }
                    }
                }
            }
        }
    }

    public void init() { // 计算一组数据的组合方式
        ArrayList array = new ArrayList(); // 创建集合数组
        for (int num1 = 0; num1 < 5; num1++) {
            for (int num2 = 0; num2 < 5; num2++) {
                if (num2 == num1)
                    continue;
                for (int num3 = 0; num3 < 5; num3++) {
                    if (num3 == num2 || num3 == num1)
                        continue;
                    for (int num4 = 0; num4 < 5; num4++) {
                        if (num4 == num3 || num4 == num2 || num4 == num1)
                            continue;
                        for (int num5 = 0; num5 < 5; num5++) {
                            if (num5 == num4 || num5 == num3 || num5 == num2
                                    || num5 == num1)
                                continue;
                            int oneArray[] = { num1, num2, num3, num4, num5 };
                            array.add(oneArray);
                        }
                    }
                }
            }
        }
        co_Array = new int[array.size()][5];    // 创建颜色的二维数组
        for (int count = 0; count < array.size(); count++) {
                                                // 循环数组初始化房子颜色数据
            co_Array[count] = (int[]) array.get(count);
        }
        pe_Array = co_Array;
        dr_Array = co_Array;
        sm_Array = co_Array;
        pet_Array = co_Array;
    }
```

【习题 20-3】编写一个程序，产生一个随机整数序列，要求不能够产生重复的随机数。

【分析】该题如果使用 Random 类，可以生成随机整数序列，但是不能够保证所生成的序列中没有重复数据出现。使用 List 来完成输出不重复的随机序列，可以使用筛选法或者是排除

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

法对生成的随机数进行判断。

【核心代码】

```
public RandomSort(int minValue, int maxValue) { // 带参数构造方法初始化变量
        this();                                  // 调用上面的构造方法
        if (maxValue >= minValue) {
            this.minValue = minValue;
            this.maxValue = maxValue;
        } else {
            System.out.println("最大值小于最小值，按默认值进行赋值！");
        }
    }
    public Integer[] exclusionSort(int len) {   // 运用排除法生成不重复的随机数序列
        if (len <= 0) {                          // 判断传入的数值
            return new Integer[0];
        } else if (len > (this.maxValue - this.minValue)) {
            System.out.println("长度不能达到:" + len + ", 长度只能是: "
                    + (this.maxValue - this.minValue));
            len = this.maxValue - this.minValue;    // 重新赋值
        }
        Random random = new Random();               // 用于生成随机对象
        List result = new ArrayList();              // 创建列表对象
        while (result.size() < len) {
    // 将[minValue, maxValue]区间等价于 min + [0, maxValue - minValue + 1) 生成随机数
            Integer randnum = new Integer(this.minValue
                    + random.nextInt(this.maxValue - this.minValue + 1));
            if (!result.contains(randnum)) {    // 判断列表中是否包含对象
                result.add(randnum);            // 添加整型对象
            }
        }
    return (Integer[]) result.toArray(new Integer[0]); // 将列表转换成整型数组返回
    }
    public Integer[] proceduresSort(int len) {  // 运用筛选法生成不重复的随机数序列
        if (len <= 0) {
            return new Integer[0];
        } else if (len > (this.maxValue - this.minValue)) {
            System.out.println("长度不能达到:" + len + ", 长度只能是: "
                    + (this.maxValue - this.minValue));
            len = this.maxValue - this.minValue;
        }
        int numLength = this.maxValue - this.minValue + 1;   // 初始化列表长度
        List list = new ArrayList();
        for (int i = this.minValue; i <= this.maxValue; i++) { // 循环依次获得整数
            list.add(new Integer(i));           // 在列表中添加整型数据
        }
        Random rd = new Random();               // 用于生成随机下标
        List result = new ArrayList();          // 创建列表对象
        while (result.size() < len) {
            int index = rd.nextInt(numLength); // 生成在[0,numLength)范围内的下标
            result.add(list.get(index));    // 下标为 index 数字对象放入列表对象中
            list.remove(index);             // 移除下标为 index 的数字对象
            numLength--;                    // 候选队列长度减去 1
        }
    return (Integer[]) result.toArray(new Integer[0]); // 将列表转换成整型数组返回
    }
    public static void printArray(Integer[] data) {      // 显示数组元素
        if (data != null) {                              // 判断数组是否为空
            for (int i = 0; i < data.length; i++) {      // 循环显示数组数据
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
                System.out.print(data[i] + "\t");
            }
        }
        System.out.println();                                    // 换行
    }
```

【习题 20-4】用 ArrayList 模拟超市，可以一次进一种商品，也可以同时进多种商品，然后将所有的商品陈列出来。

【分析】主要考查 List 的添加与遍历。

【核心代码】

```
public class SuperMarket {
    public static void main(String[] args) {
        List<String> storage = new ArrayList<String>();    // 超市仓库
        storage.add("foods");
        storage.add("tools");
        storage.add("books");
        storage.add("cars");
        List<String> clothing = new ArrayList<String>();   // 大量进货
        clothing.add("dress");
        clothing.add("costume");
        clothing.add("gown");
        storage.addAll(clothing);
        System.out.println("超市的商品: ");
        for (String p : storage) {
            System.out.println(p);
        }

        System.out.println("\n 调整商品结构");
        storage.remove("cars");                                  // 不卖汽车了
        System.out.println("更新后的商品: ");
        for (String p : storage) {
            System.out.println(p);
        }
    }
}
```

【习题 20-5】使用 ListIterator 模拟警察调度公路上的车辆，可以查看来的方向或去的方向是否有车，而且可以随意将某车加入到任意位置或移下公路。

【分析】主要考查 ListIterator 添加删除和遍历元素。

【核心代码】

```
public class Road {
    public static void main(String[] args) {
        List<String> cars = new ArrayList<String>();
        cars.add("Ford");
        cars.add("Dodge");
        cars.add("citroen");
        cars.add("Honda");
        cars.add("Hummer");
        cars.add("Jeep");
        cars.add("Mazda");
        ListIterator<String> li = cars.listIterator(); // 取得“警察”对象
        System.out.println("放行: "+ li.next());
        System.out.println("违章，靠边停车: "+ li.next());
        li.remove();
        String cc = li.next();
        System.out.println("让"+cc+"到前面去");
        li.remove();
        li.next();
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
            li.next();
            li.next();
            li.set(cc);
            System.out.println("\n 现在道路畅通了");
            for (String c : cars) {
                System.out.println(c);
            }
        }
}
```

【习题 20-6】使用 LinkList 模拟队列（queue）操作。

【分析】队列就是先进去的元素，先出来。

【核心代码】

```
public class Queue {
    private LinkedList<String> queue = new LinkedList<String>();
    public void insert(String s) {
        queue.addLast(s);
    }
    public String remove() {
        return queue.removeFirst();
    }

    public static void main(String[] args) {
        Queue q = new Queue();
        q.insert("Mike");
        q.insert("Jack");
        q.insert("Rose");
        System.out.println(q.remove());
        System.out.println(q.remove());
        System.out.println(q.remove());
    }
}
```

【习题 20-7】使用 LinkList 模拟堆栈（stack）操作。

【分析】堆栈就是最先进去的元素，最后出来。

【核心代码】

```
public class Stack {
    private LinkedList<String> stack = new LinkedList<String>();
    public void push(String s) {
        stack.addFirst(s);
    }
    public String pop() {
        return stack.removeFirst();
    }

    public static void main(String[] args) {
        Stack s = new Stack();
        s.push("Peugeot");
        s.push("Volvo");
        s.push("Lamborghini");

        System.out.println(s.pop());
        System.out.println(s.pop());
        System.out.println(s.pop());
    }
}
```

【习题 20-8】编写一个程序模拟一个教室的座位，每个学生可以有而且只能有一个座位。

【分析】考查 HashSet 不能有重复元素的特性。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

【核心代码】

```
public class Room {
    public static void main(String[] args) {
        HashSet<String> seats = new HashSet<String>();
        seats.add("Jack");
        seats.add("Mike");
        seats.add("Jerry");
        seats.add("Troy");
        seats.add("Harry");
        for (String seat : seats) {
            System.out.println(seat);
        }

        System.out.println("Troy 想要两个座位，没有得到");
        seats.add("Troy");
        for (String seat : seats) {
            System.out.println(seat);
        }
    }
}
```

【习题 20-9】编写一个程序模拟公司员工信息，公司允许有姓名和薪水相同的员工。

【分析】考查 ArrayList 的使用。

【核心代码】

```
public class Employee {
    private String name;
    private int salary;

    public Employee(String name, int salary) {
        this.name = name;
        this.salary = salary;
    }

    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }

    public int getSalary() {
        return salary;
    }
    public void setSalary(int salary) {
        this.salary = salary;
    }
}

public class Company {
    public static void main(String[] args) {
        List<Employee> es = new ArrayList<Employee>();
        es.add(new Employee("Jack", 10000));
        es.add(new Employee("Rose", 12000));
        es.add(new Employee("Jerry", 8000));
        es.add(new Employee("Mike", 10000));
        es.add(new Employee("Tom", 6000));
        for (Employee e : es) {
            System.out.println(e.getName() + " 的薪水是: "+e.getSalary());
        }
    }
}
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

【习题 20-10】编写一个程序模拟班级学生的成绩库，每个学生对应一个成绩，实现动态删除和更改学生成绩，并且可以打印所有学生的成绩。

【分析】主要考查 HashMap 的应用。

【核心代码】

```
public class GradeList {
   public static void main(String[] args) {
      HashMap<String, Integer> grades = new HashMap<String, Integer>();
      grades.put("Tom", 70);
      grades.put("Jack", 65);
      grades.put("Mike", 76);
      grades.put("Rose", 90);
      for (Map.Entry e : grades.entrySet()) {
         System.out.println(e.getKey() + "的成绩是："+ e.getValue());
      }
      System.out.println("\n 修改后的成绩：");
      grades.put("Jack", 80);
      grades.put("Rose", 87);
      for (Map.Entry e : grades.entrySet()) {
         System.out.println(e.getKey() + "的成绩是："+ e.getValue());
      }
   }
}
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第 21 章　泛　　型

泛型是在 Java SE 5.0 才开始引入的一个新特性，使用泛型可以降低在开发中出现错误的机会，减少代码量，可以使程序更加安全，更具有可读性。在本章中将主要介绍泛型程序的开发。



21.1　泛型概述

所谓泛型，就是将类型也作为一种参数来进行指定，在使用泛型后，类型就不再是一成不变的。如果不使用泛型，代码中的类型则是固定不变的，这样就会为开发带来很大的不便，例如以下代码。

```
01  //创建一个用来存放 Integer 对象的列表
02  List MyList=new ArrayList();
03  //向列表中添加一个 Integer 对象
04  MyList.add(new Integer(0));
05  //从列表中取出 Integer 对象
06  Integer x=(Integer)MyList.get(0);
```

在以上代码中创建列表的时候不可以告诉系统列表将要存放何种类型的对象，开发人员需要注意，不能够将其他的类型对象添加到列表中，否则会降低安全性。如果在其中添加了其他类型的对象，可能会出现程序的运行不正常等错误。

在使用了泛型之后，类型就可以作为一种参数进行指定，例如以下代码。

```
01  //创建泛型指定元素的类型的列表
02  List<Integer> MyList=new ArrayList<Integer>();
03  //向列表中添加一个元素
04  MyList.add(new Integer(0));
05  //从列表中取出 Integer 对象
06  Integer x=MyList.get(0);
```

在上面的代码中使用了泛型，也就是在第 02 行中通过泛型的语法指定了列表中的元素类型，“<Integer>”就是通过泛型参数告诉系统该列表中只能存放 Integer 类型的元素。而在读取列表中的元素时，也不需要进行强制转换了。



21.2　泛型格式

在前面介绍了泛型的基本概念和作用，本节就主要介绍泛型在开发方面的相关知识，主要包括泛型类或者是接口的声明泛型方法的开发以及类型变量的限制等。

21.2.1　定义泛型类

泛型类的声明和普通类的声明没有太大的区别，前者只是在类的声明时加入一个或者多个变量声明，下面是泛型类的基本语法。

```
public class 类名<类型的变量>
{
实体
}
```

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

对于泛型类的类型变量，需要使用一对尖括号（<>）将其包围，并且要放置在类的后面。尖括号中的类型变量可以是任何符合 Java 命名规范的标识符，通常都在其中使用单个的大写字母。例如，使用 E 来表示集合元素的类型，K 和 V 则表示键值对中的键类型和值类型，T、U、S 表示为任意类型。泛型类中的类变量可以有多个，就像方法的参数一样，多个类型变量之间可以使用逗号（,）来隔开。

【范例 21-1】示例代码 21-1 是一个使用泛型的示例。

示例代码 21-1

```
01  class MyValue<T,U>
02  {
03      //定义两个由类型变量指定类型的成员变量
04      private T value1;
05      private U value2;
06      //定义两个成员变量的 set 方法
07      public void setValue(T value1,U value2)
08      {
09          this.value1 = value1;
10          this.value2 = value2;
11      }
12      //定义两个成员变量的 get 方法
13      public U getValue2()
14      {
15          return value2;
16      }
17      public T getValue1()
18      {
19          return value1;
20      }
21  }
22  public class MyValueTest
23  {
24      public static void main(String[] args)
25      {
26          //创建自定义的泛型类型对象，并且指定两个类型变量
27          MyValue<String,Integer> mv1=new MyValue<String,Integer>();
28          //调用 set 方法对成员变量进行设置
29          mv1.setValue("该参数为字符串类型",new Integer(100));
30          //返回第一个成员变量的值
31          String str1=mv1.getValue1();
32          //返回第二个成员变量的值
33          Integer int1=mv1.getValue2();
34          //打印结果
35      System.out.println("第一个参数对象的值为："+str1+"\n 第二个值为："+int1+"。\n");
36          //创建另一个自定义泛型类型对象，并且指定两个类型变量
37          MyValue<Integer,String> mv2=new MyValue<Integer,String>();
38          //调用 set 方法对成员变量进行设置
39          mv2.setValue(new Integer(100),"该参数为字符串类型");
40          //返回第一个成员变量的值
41          String str2=mv2.getValue2();
42          //返回第二个成员变量的值
43          Integer int2=mv2.getValue1();
44          //打印结果
45      System.out.println("第一个参数对象的值为："+int2+"\n 第二个值为："+str2+"。");
37      }
38  }
```

【运行结果】程序运行结果如下。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
第一个参数对象的值为：该参数为字符串类型
第二个值为：100。

第一个参数对象的值为：100
第二个值为：该参数为字符串类型。
```

【代码解析】以上代码中，首先定义了一个泛型类 MyValue，其中有类型参数 T 和 U，并设置两个成员变量，然后设置成员变量的 set 和 get 方法（第 01～21 行）。最后在测试类中创建 MyValue 的对象，并且将类型参数 T 指定为 String 类型，参数 U 指定为 Integer（第 27～29 行）。

通过泛型类的类型参数可以实现方法的泛型化。Java 中也支持单独对一个方法使用泛型，也就是说，方法也可以带有一个或者多个类型参数，语法如下。

```
public static <类型变量> 返回类型 方法名（参数）
{
方法体
}
```

对于泛型的方法来说，可以在普通的类中进行定义，也可以在泛型类中进行定义，泛型方法的类型变量也是在“<>”中进行声明的，同样也可以有很多个类型变量，也是使用“,”隔开。但是它与泛型类不同的是，泛型方法的类型参数声明需要放在方法的返回类型声明之前，而不是像泛型类一样放到类声明的后面。

【范例 21-2】示例代码 21-2 是一个使用泛型示例。

示例代码 21-2

```
01  public class FFTest
02  {
03      //自定义泛型方法
04      public static <T> T getLast(T[] a)
05      {
06          //返回指定类型数组的最后一个元素。
07          return a[a.length-1];
08      }
09      public static void main(String[] args)
10      {
11          //创建字符串类型的数组
12          String[] str={"第一个元素","第二个元素","第三个元素"};
13          //使用泛型方法，并为其指定类型参数
14          String laststr=FFTest.<String>getLast(str);
15          //打印结果
16          System.out.println("String 数组 str 的最后一个元素是："+laststr);
17          //创建 Integer 类型数组
18          Integer[] inte={new Integer(100),new Integer(200),new Integer(300)};
19          //使用泛型方法，但是没有显示指定类型的参数
20          Integer lastint=FFTest.getLast(inte);
21          //打印结果
22          System.out.println("Integer 数组 inte 的最后一个元素是："+lastint);
23      }
24  }
```

【运行结果】程序运行结果如下。

```
String 数组 str 的最后一个元素是：第三个元素
Integer 数组 inte 的最后一个元素是：300
```

【代码解析】首先定义了泛型的方法 getLast，并设置返回数组的最后一个元素（第 04～08

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

行)。在主方法中定义了两个不同类型的数组，然后分别代入该方法，并且在第一次调用的时候给出参数的类型，然后打印输出（第 12～22 行)。

21.2.2 类型变量的限制

如果希望类型变量中的类型有一定的范围，而不是任意类型都可以，这样的情况可以通过使用类型变量的限制来实现，其语法格式如下。

```
类型变量 extends 限定类型序列
```

限定类型序列是表示类型变量可以指定的类型范围，例如“T extends Jcomponent”表示类型变量 T 只能为 Jcomponent 或者其子类。该序列可以是类或者接口，多个类之间使用“&”进行连接。

【范例 21-3】示例代码 21-3 是一个使用泛型示例。

示例代码 21-3

```
01  public class ExtendsTest
02  {
03      public static void main(String[] args)
04      {
05          //创建 ValueTest 的对象，并指定类型参数为 Integer
06          ValueTest<Integer> vt=new ValueTest<Integer>();
07          //设置 vt 中的参数
08          vt.setValue(new Integer(200));
09          //获取 vt 中的值
10          Integer inte=vt.getValue();
11          //打印输出
12          System.out.println("vt 对象中的值是："+inte+
13                  "\n 该值既属于 Number，又实现了 Comparable 和 Serializable");
14      }
15  }
16  class ValueTest <T extends Number & Comparable &Serializable>
17  {
18      //定义由类型参数指定类型的成员变量
19      private T value;
20      //设置成员变量的 set 方法
21      public T getValue()
22      {
23          return value;
24      }
25      //设置成员变量的 get 方法
26      public void setValue(T value)
27      {
28          this.value = value;
29      }
30  }
```

【运行结果】程序运行结果如下。

```
vt 对象中的值是：200
该值既属于 Number，又实现了 Comparable 和 Serializable
```

【代码解析】以上程序中，使用类型变量来限制指定类型变量的类型范围，必须是 Number 的子类，而且要同时实现了 Comparable 和 Serializable 接口（第 16 行)。在主方法中指定类型变量的类型为 Integer，并且满足类型范围的要求（第 06 行)。最后为对象设置值，并打印输出（第 08～13 行)。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



21.3　泛型参数的继承和通配符

在前面只是介绍了简单的泛型的使用方法，而在泛型中，还有许多比较复杂的知识，例如泛型的继承、泛型通配符等。在本节中，将主要介绍这两个知识点。

21.3.1　泛型的继承

前面几章中介绍了继承问题。使用继承可以带来多态、重用等一些优点。由于泛型参数是用来表示类型的，因此，也会涉及继承。不过需要注意的是，继承规则和直观的想象是不同的，例如以下代码。

```
01  public class InheritTest
02  {
03      public static void main(String[] args)
04      {
05          //声明一个创建类型参数为 String 的列表
06          ArrayList<String> as=new ArrayList<String>();
07          //声明类型参数为 Object 的列表引用
08          ArrayList<Object> os=new ArrayList<Object>();
09          //对类型参数为 Object 的列表引用赋值
10          os=as;
11      }
12  }
```

在上面的代码中，分别设置了两个不同类型参数的列表，然后将 as 赋值给 os，这一步看似是没有错误的。但是在实际运行中却会抛出“不兼容的类型”错误。因为直观感觉上“ArrayList<String>”可以看做是“ArrayList<Object>”，但实际上不是这样的，由于“ArrayList<Object>”中可以放入任何类型的对象，而“ArrayList<String>”则只能够存放字符串。所以，ArrayList<String>不可以被当做“ArrayList<Object>”来使用。

21.3.2　泛型中的通配符

泛型的继承和普通类的继承是不同的，如果 A 类和 B 类是父类和子类的关系，则 K<A>并不可以看做是 G，如果想要达到如此的效果，则需要使用泛型的通配符“？”。例如以下代码。

【范例 21-4】示例代码 21-4 是一个使用通配符示例。

示例代码 21-4

```
01  public class WildcardTest
02  {
03      public static void printMyValue(ArrayList<?> mv)
04      {
05          //打印该对象中的值
06          System.out.println("列表中的值是：");
07          System.out.println(mv);
08      }
09      public static void main(String[] args) {
10          //创建类型参数 String 的 ArrayList 对象
11          ArrayList<String> as=new ArrayList<String>();
12          //循环插入 20 个字符串
13          for(int i=0;i<20;i++)
14          {
15              as.add(" "+i);
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
16            }
17            //调用 printMyValue 方法进行打印
18            printMyValue(as);
19        }
20    }
```

【运行结果】程序运行结果如下。

```
列表中的值是:
[ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9,  10,  11,  12,  13,  14,  15,  16,  17,  18,
19]
```

【代码解析】以上程序中，在 printMyValue 方法的参数中使用了通配符，“ArrayList<?>”则表示可以接收任意具体类型的参数（第 03～07 行）。然后创建类型参数为 String 的 ArrayList 对象，表示该列表中只能存放字符串，并循环向其中添加了 20 个字符串（第 11～16 行）。最后调用 printMyValue 方法进行打印输出（第 18 行）。

使用通配符需要注意以下两点。

- 泛型通配符只能够用于引用的声明中，不可以在创建对象的时候使用。
- 不能使用采用了泛型通配符的引用来调用使用了泛型参数的方法。

21.4 习题

【习题 21-1】使用 String 作为 ArrayList 的类型参数，声明一个 List，并添加元素和遍历所有的元素。

【分析】主要考查泛型的使用。

【核心代码】

```
public class Sentence{
   public static void main(String[] args) {
      List<String> ls = new ArrayList<String>();
      ls.add("日照香炉生紫烟");
      ls.add("遥看瀑布挂前川");
      ls.add("飞流直下三千尺");
      ls.add("疑是银河落九天");
      for (String s : ls) {
         System.out.println(s);
      }
   }
}
```

【习题 21-2】使用 Student 作为 ArrayList 的类型参数，声明一个 List，并进行添加元素和遍历所有元素的操作。

【分析】主要考查泛型的使用。

【核心代码】

```
public class Student {
   private String name;
   private Integer score;

   public Student(String name, Integer score) {
      this.name = name;
      this.score = score;
   }

   public static void main(String[] args) {
      List<Student> ss = new ArrayList<Student>();
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
        ss.add(new Student("Jack", 60));
        ss.add(new Student("Tom", 80));
        ss.add(new Student("Rose", 70));
        ss.add(new Student("Mike", 50));
        for (Student s : ss) {
            System.out.println(s.name + "的成绩: "+s.score);
        }
    }
}
```

【习题 21-3】使用 String 作为 HashMap 的类型参数，声明一个 Map，并添加元素和遍历所有的元素。

【分析】主要考查泛型的使用。

【核心代码】

```
public class Nationality{
    public static void main(String[] args) {
        Map<String, String> map = new HashMap<String, String>();
        map.put("Jack", "China");
        map.put("Tom", "USA");
        map.put("Rose", "France");
        map.put("Mike", "UK");
        map.put("Peter", "Germany");
        for (String s : map.keySet()) {
            System.out.println(s + "来自" + map.get(s));
        }
    }
}
```

【习题 21-4】编写一个 Worker 类，使用双类型参数，分别表示工作、吃饭和薪水。

【分析】主要考查泛型的使用。

【核心代码】

```
public class Worker<T, S> {
    private T first;
    private S second;

    public Worker(T first, S second) {
        this.first = first;
        this.second = second;
    }

    public static void main(String[] args) {
        Worker<String, String> w1 = new Worker<String, String>("iron", "water");
        Worker<String, Integer> w2 = new Worker<String, Integer>("foods", 30);
        Worker<Integer, String> w3 = new Worker<Integer, String>(10000, "doller");
        System.out.println("worker1 的工作原料"+ w1.first + " 和 " + w1.second);
        System.out.println("worker2 的午饭吃"+ w2.first + " " + w2.second + "碗");
        System.out.println("worker3 的薪水"+ w3.first + " " + w3.second);
    }
}
```

【习题 21-5】使用 String 和 Integer 作为 HashMap 的泛型参数，分别表示学生的名字和成绩。

【分析】主要考查泛型的使用。

【核心代码】

```
public class Score {
    public static void main(String[] args) {
        Map<String, Integer> scores = new HashMap<String, Integer>();
        scores.put("Jack", 70);
        scores.put("Tom", 80);
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
        scores.put("Peter", 78);
        scores.put("Mike", 83);
        for (String s : scores.keySet()) {
            System.out.println(s + "的成绩: " + scores.get(s));
        }
    }
}
```

【习题 21-6】编写一个税费类，分别表示少到几毛钱，多到几十亿元的钱。

【分析】主要考查泛型的使用。

【核心代码】

```
public class Tax<T extends Number> {
    private T value;

    public Tax(T v) {
        value = v;
    }
    public static void main(String[] args) {
        Tax<Float> shop = new Tax<Float>(366.5f);
        Tax<Double> superMarket = new Tax<Double>(5005d);
        Tax<Integer> chinaTele = new Tax<Integer>(4000000);
        Tax<Long> sinopetro = new Tax<Long>(300000000001);

        System.out.println("小商店今年纳税: "+shop.value);
        System.out.println("大超市今年纳税: "+superMarket.value);
        System.out.println("中国电信今年纳税: "+chinaTele.value);
        System.out.println("中国石油今年纳税: "+sinopetro.value);
    }
}
```

【习题 21-7】使用 String 和 Integer 作为 Map 的类型参数，表示汽车的价格，并添加元素和遍历所有的元素。

【分析】主要考查泛型的使用。

【核心代码】

```
public class Car {
    public static void main(String[] args) {
        Map<String, Integer> prices = new HashMap<String, Integer>();
        prices.put("Lincoln", 7000000);
        prices.put("Lexus", 200000);
        prices.put("Ferrari", 5000000);
        prices.put("Ford", 220000);
        for (String s : prices.keySet()) {
            System.out.println(s + "的价格: " + prices.get(s));
        }
    }
}
```

【习题 21-8】编写一个仓库类，这个仓库可以存储不同类型公司的不同类型的货物。

【分析】主要考查泛型的使用。

【核心代码】

```
public class Store<T, S> {
    private T product;
    private S company;

    public Store(T product, S company) {
        this.product = product;
        this.company = company;
    }
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
    public static void main(String[] args) {
        Store<String, Integer> s1 = new Store<String, Integer>("apple", 100);
        Store<Integer, Integer> s2 = new Store<Integer, Integer>(500, 100);
        Store<String, String> s3 = new Store<String, String>("radio", "sony");
        System.out.println("company: "+ s1.company + ", product:"+ s1.product);
        System.out.println("company: "+ s2.company + ", product:"+ s2.product);
        System.out.println("company: "+ s3.company + ", product:"+ s3.product);
    }
}
```

【习题 21-9】使用 Company 作为 List 的类型参数，并添加和遍历元素。

【分析】主要考查泛型的使用。

【核心代码】

```
public class Company {
    private String name;
    private Number revenue;

    public Company(String name, Number revenue) {
        this.name = name;
        this.revenue = revenue;
    }

    public static void main(String[] args) {
        List<Company> cs = new ArrayList<Company>();
        cs.add(new Company("chinaTele", 5000000));
        cs.add(new Company("sinopetro", 300000000001));
        cs.add(new Company("chinaMobile", 6000000));
        for (Company c : cs) {
            System.out.println(c.name+ "总收入: "+ c.revenue);
        }
    }
}
```

【习题 21-10】在 List 中使用通配符泛型，用不同的类型作为泛型参数声明 List，然后打印出这个 List。

【分析】主要考查通配符泛型的使用。

【核心代码】

```
public class WildCard {
    public static void print(List<?> store) {
        System.out.println(store);
    }

    public static void main(String[] args) {
        List<String> ss = new ArrayList<String>();
        ss.add("Jack");
        ss.add("Mike");
        ss.add("Rose");
        print(ss);
        List<Integer> is = new ArrayList<Integer>();
        is.add(400);
        is.add(700);
        is.add(900);
        is.add(1200);
        print(is);
    }
}
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第 22 章　数 据 结 构

数据结构（Data Structure）是计算机科学的重要研究领域之一，主要研究计算机操作对象以及对象间的关系和处理方法。



22.1　数据结构的基本概念

数据结构的概念包括：数据、数据元素、数据项。数据是计算机处理和加工的对象，是信息的载体，是数据元素的集合。数据元素是数据的基本单位，由一个或多个数据项组成。数据项是具有实际意义的最小单位。

数据结构是数据的组织结构，分为物理结构和逻辑结构。所谓物理结构，就是数据在物理内存中的存储形式，是以电子信号的方式存在的。而逻辑结构主要是指数据之间的逻辑关系，表示方法和处理算法等一系列问题。根据数据元素之间关系的不同，数据结构可以分为线性结构、树形结构、图形结构、集合结构。线性结构简单易用，是最常用的一种数据结构。

通过对数据结构的学习，读者能够学会分析数据的基本特征，掌握组织数据和处理数据的基本算法。



22.2　算法

数据结构描述的是数据的组织形式，算法（Algorithm）描述的是数据的处理方法。数据结构和算法是程序设计的基础。

22.2.1　算法概述

算法描述解决问题的策略，是一系列指令的集合，对一定规范的输入能够在一定的时间内输出。算法具有以下五个特征。

- 输入：即一个算法需要描述运算对象初始状况的数据。
- 输出：即一个算法必须输出对输入数据的加工结果，没有结果的算法没有意义。
- 有穷性：即一个算法必须在执行完有限的指令后终止。
- 确切性：即一个算法的每个步骤必须有确切的意义。
- 可行性：即一个算法的任何步骤都可以被分解为人工可执行的运算。

22.2.2　算法分析

算法分析的目的在于选择合适的算法和改进算法。对同一个问题，可以采用不同的算法，不同的算法可能需要不同的时间和存储空间来完成。评价一个算法，可以从两个不同的维度来考虑。

- 时间维度：从执行算法所需要的时间考虑。一般地，算法是问题的规模 n 的函数 $f(n)$，算法消耗的时间可以表示为 $T(n)=O(f(n))$，称 $O(f(n))$为算法的时间复杂度。
- 空间维度：从执行算法所需要的内存空间考虑。与时间复杂度类似，把 $S(n)=O(f(n))$称

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



为算法的空间复杂度。

分析算法可以预测某一算法适合什么样的程序环境，在什么样的环境下更有效地运行，对解决同样问题的不同算法的有效性进行比较。

22.2.3　递归算法

递归可以使复杂的问题变得简单，使程序变得更加简洁和清晰。递归算法的基本思想就是：将规模较大、解决较困难的问题转化成为规模较小、解决相对较容易的相同问题；规模较小的问题又转化为更小规模的问题，直到小到可以直接得出解为止，从而使整个问题得到解决。

简单地说，递归就是程序方法直接或间接调用其自身的算法，汉诺塔问题就是典型的递归算法。汉诺塔问题源自一个古老的故事：神创造世界时，造了三座塔，A 塔从下往上，按从大到小的顺序放着 64 个金盘子，神命令婆罗门把所有的金盘子移动到 C 塔上，并且盘子还是从下往上，按从大到小的顺序排列，并且规定，一次只能移动一个盘子，大盘子不能放在小盘子上，可以借助 B 塔。现在做如下分析：

- 开始时，A 塔从下到上放着很多盘子，顺序是由大到小。
- 任务是把所有的盘子移到 C 塔，顺序不变，可以借助 B 塔。
- 要求每次移动一个盘子，并且大盘子必须在小盘子下面。
- 使用数学算法知道，当有 n 个盘子时，需要移动 $f(n)=2^n-1$ 次。

上面对汉诺塔问题做了简单分析，下面就用程序实现汉诺塔算法，将盘子从 A 移动到 C，中间借助 B。

【范例 22-1】汉诺塔算法程序实现。

示例代码 22-1

```
01  public class Hanoi {
02     private static int i = 0;
03     public static void main(String[] args) {
04        int n = 0;
05        Scanner reader = new Scanner(System.in); // 扫描输入文本
06        System.out.println("请输入金盘子的数量：");
07        n = reader.nextInt();
08        System.out.println(n +" 层汉诺塔的解法是：");
09        // 将 n 个盘子从 A 移动到 C，中间借助 B
10        move(n, 'A', 'B', 'C');
11        System.out.println("Total: "+ i);
12        System.exit(0);
13     }
14     // 移动盘子的方法，将 n 个盘子从 from 移到 to，inter 辅助
15     public static void move(int no, char from, char inter, char to) {
16        i++;
17        if (no == 1) {
18           System.out.println("Move plate 1 from "+ from + " to "+ to);
19        } else {
20           move(no-1, from, to, inter); // 先将 n-1 个盘子从 from 移动到 inter
21           System.out.println("Move plate " + no +" from " + from +" to "+ to);
22           move(no-1, inter, from, to); // 再将 n-1 个盘子从 inter 移动到 to
23        }
24     }
25  }
```

【运行结果】程序运行结果如下。

```
请输入金盘子的数量：
3
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
3 层汉诺塔的解法是:
Move plate 1 from A to C
Move plate 2 from A to B
Move plate 1 from C to B
Move plate 3 from A to C
Move plate 1 from B to A
Move plate 2 from B to C
Move plate 1 from A to C
Total: 7
```

【代码解析】在以上程序中，首先读取盘子的数量（第 05～08 行），然后调用移动盘子的方法 move()。在 move()方法中，如果是最后一个盘子，就简单移动到 C（第 17、18 行）；如果不是，就迭代调用 move()方法，先将第 n–1 个盘子移动到 B，再从 B 移动到 C（第 19～23 行）。要运用递归，就必须满足以下条件。

- 原问题可以层层转化为规模越来越小的子问题，即把求 n!转化为求(n–1)!，再转为求(n–2)!…
- 对于规模足够小的问题，能够直接得解，即对于 0!，能够直接得到结果 1。

22.2.4 排序算法

排序是程序设计中广泛应用的技术之一，比如，超市查出价格最便宜的 100 种商品。

所谓排序，就是将一组元素按指定关键字的升序或降序进行重新排列的算法。

冒泡排序是比较常用的一种排序方法。

所谓冒泡排序，就是把一组元素中的每个元素看做气泡，根据轻气泡不能在重气泡之下的原则，从下往上查看这组元素，凡有不符合此原则的气泡就向上漂浮，如此反复进行，直到任何两个气泡比较，都是轻者在上、重者在下为止。

冒泡排序的基本方法就是：两两比较待排序元素，如果两个元素的次序相反，就进行交换，直到没有相反的顺序为止。

设想在数组 A 中有 n 个数，要求把这 n 个数从大到小排序，其基本过程如下：

① 在 A[0]～A[n–1]范围内，依次比较两个相邻元素的值，若 A[i]<A[i+1]，则交换 A[i]与 A[i+1]的位置，经过这一趟冒泡，就把数组中最小的数放到了 A[n–1]的位置。

② 再对 A[0]～A[n–2]范围内的数进行冒泡比较，将该范围内最小的数放到 A[n–2]的位置。

③ 依次进行，最多需要 n–1 趟冒泡比较，即可完成排序。

④ 如果在某趟冒泡过程中，没有交换相邻的数，则说明冒泡可以结束。

【范例 22–2】冒泡排序算法程序实现。

示例代码 22-2

```
01  public class Bubble {
02     public static void main(String[] args) {
03        Scanner reader = new Scanner(System.in); // 扫描输入文本
04        System.out.println("请输入待排序数组（以空格分隔）: ");
05        String st = reader.nextLine();
06        System.out.println(st + " 的排序方法: ");
07        String[] ss = st.split("\\s+");
08        sort(ss); // 排序
09        System.out.println("结果: ");
10        StringBuffer sb = new StringBuffer();
11        for (String a : ss) {
12           sb.append(a).append(" ");
13        }
14        System.out.println(""+ sb);
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
    }
    public static <E extends Comparable<E>> void sort(E[] array) {
        StringBuffer sb = new StringBuffer();
        int len = array.length;
        for (int i = len - 1; i >= 0; i--) {
            for (int j = 0; j < i; j++) {
                sb.append("比较：").append(array[j]).append(" ").append(array[j+
1]);
                if (array[j].compareTo(array[j + 1]) > 0) {
                   sb.append(" 冒泡");
                    swap(array, j, j+1);
                }
                System.out.println(sb);
                sb.setLength(0);
            }
            for (E s : array) {
                sb.append(s).append(" ");
            }
            System.out.println("第 "+ (len - i) + " 轮："+ sb);
            sb.setLength(0);
        }
        System.out.println("-----------");
    }
    // 交换两个元素的位置
    public static <E extends Comparable<E>> void swap(E[] array, int a, int b)
{
        E t = array[a];
        array[a] = array[b];
        array[b] = t;
    }
}
```

【运行结果】程序运行结果如下。

```
请输入待排序数组（以空格分隔）：
3 9 4 7 2 5
3 9 4 7 2 5 的排序方法：
比较：3 9
比较：9 4 冒泡
比较：9 7 冒泡
比较：9 2 冒泡
比较：9 5 冒泡
第 1 轮：3 4 7 2 5 9
比较：3 4
比较：4 7
比较：7 2 冒泡
比较：7 5 冒泡
第 2 轮：3 4 2 5 7 9
比较：3 4
比较：4 2 冒泡
比较：4 5
第 3 轮：3 2 4 5 7 9
比较：3 2 冒泡
比较：3 4
第 4 轮：2 3 4 5 7 9
比较：2 3
第 5 轮：2 3 4 5 7 9
第 6 轮：2 3 4 5 7 9
-----------
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

结果：
2 3 4 5 7 9

【代码解析】在以上程序中，首先读取待排序的数字，并且分隔成数组（第 03～07 行），然后调用 sort()方法进行排序。在 sort()方法中，先两两比较 0～n–1 之间相邻的数字，如果前一个数字比后一个大，就交换位置，再比较 0～n–2 之间的数字，以此类推。



22.3 链表

链表是一种非常重要的线性表，其通过为每个节点提供一个额外指针域，保存相邻节点的内存地址，实现各个节点相互连接。

链表的特点是可以动态地改变长度，非常适合动态地对元素进行添加和删除操作。由于需要逐点比较进行查询，所以随机访问和遍历的速度就比较慢。根据节点中指针的数目和连接方式，链表可以分为：单链表、双链表、循环链表。

- 单链表：节点由两部分组成，一是此节点的数据，二是指向后面节点的指针。
- 双链表：在单链表的基础上，为每个节点增加指向前面节点的指针。
- 循环链表：改变单链表和双链表的头节点和尾节点，使之分别指向尾节点和头节点，形成一个循环。从循环表的任意节点出发都能访问到其他节点。

22.3.1 Java 中的链表

链表通过指针将节点串联在一起，Java 中没有指针，但可以通过对象引用来实现节点间的连接。

LinkedList 是一个典型的 Java 实现的双向链表，每个节点除了数据本身外，还有两个引用分别指向前一个节点和后一个节点。下面以 LinkedList 为例讲解链表的添加和删除操作。

22.3.2 链表的添加

LinkedList 提供的添加元素的方法如下：

- public boolean add(E e)：将元素 e 添加到链表的末尾。
- public void add(int index, E e)：将元素 e 添加到指定的位置，此位置上的元素和其他后续元素都向右移动（索引增加 1）。
- public Boolean addAll(Collection<? extends E> c)：添加集合 c 中的所有元素到链表的末尾，按集合 c 的迭代器（iterator）返回元素的顺序。操作过程中，如果此集合 c 被修改，则出现错误。
- public Boolean addAll(int index, Collection c)：在指定位置将集合 c 中的所有元素添加到链表。将当前位置的元素及后续元素都向右移动（增加索引）。新元素按照集合 c 迭代器（iterator）返回的顺序排列。
- public void addFirst(E e)：将元素 e 添加到链表开头。
- public void addLast(E e)：将元素 e 添加到链表末尾。

【范例 22-3】演示链表的添加操作。

示例代码 22-3

```
01  public class LinkedListAdd {
02      private static LinkedList<String> link = new LinkedList<String>();
03      public static void main(String[] args) {
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



```
04          link.add("a");
05          link.add("b");
06          link.add("c");
07          echo();
08          link.add(0, "Make the index 0");
09          echo();
10          link.addFirst("This is the first.");
11          echo();
12          link.addLast("This is the last.");
13          echo();
14          link.addAll(Arrays.asList("1", "2", "3"));
15          echo();
16          link.addAll(0, Arrays.asList("A", "B", "C"));
17          echo();
18      }
19      public static void echo() {
20          for (int i = 0; i < link.size(); i++) {
21              System.out.println(i + " : " + link.get(i));
22          }
23          System.out.println("-------------------------");
24      }
25  }
```

【运行结果】程序运行结果如下。

```
0 : a
1 : b
2 : c
-------------------------
0 : Make the index 0
1 : a
2 : b
3 : c
-------------------------
0 : This is the first.
1 : Make the index 0
2 : a
3 : b
4 : c
-------------------------
0 : This is the first.
1 : Make the index 0
2 : a
3 : b
4 : c
5 : This is the last.
-------------------------
0 : This is the first.
1 : Make the index 0
2 : a
3 : b
4 : c
5 : This is the last.
6 : 1
7 : 2
8 : 3
-------------------------
0 : A
1 : B
2 : C
3 : This is the first.
4 : Make the index 0
5 : a
6 : b
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
7 : c
8 : This is the last.
9 : 1
10 : 2
11 : 3
-------------------------
```

【代码解析】在以上程序中，首先实例化了一个 LinkedList 对象，然后在 main()方法中向这个链表添加元素。第 04～06 行按顺序添加元素，第 10 行将一个字符串添加到指定位置，第 10 行和第 12 行分别将字符串添加到第一个和最后一个位置，第 14 行将列表添加到链表中，第 16 行将列表添加到链表的指定位置。

22.3.3 链表的删除

LinkedList 提供的删除元素的方法如下：

- public E remove()：返回并移除链表的头（第一个元素）。
- public E remove(int index)：返回并移除指定索引的元素，将后续元素向左移（索引减 1）。
- public Boolean remove(Object o)：移除链表中第一次出现的指定对象 o。如果链表中不存在此元素，则不改变。更准确地讲，移除满足(o == null ? get(i) == null : o.equals(get(i)))条件的具有最小索引的元素（如果存在这样的元素）。
- public E removeFirst()：返回并移除链表的头元素。
- public Boolean removeFirstOccurrence(Object o)：移除链表中第一次出现的指定对象 o（从头到尾遍历）。
- public E removeLast()：返回并移除链表的最后一个元素。
- public Boolean removeLastOccurrence(Object o)：移除链表中最后一次出现的指定对象 o（从头到尾遍历）。

【范例 22-4】演示链表删除操作的程序。

示例代码 22-4

```
01  public class LinkedListRemove {
02      private static LinkedList<String> link = new LinkedList<String>();
03
04      public static void main(String[] args) {
05          link.addAll(0, Arrays.asList(
06                  "First.",
07                  "Second",
08                  "Third",
09                  "Fourth",
10                  "Fifth",
11                  "Fifth",
12                  "Sixth",
13                  "Sixth"
14          ));
15          echo("所有的元素");
16          link.remove();
17          echo("执行 remove() 后");
18          link.remove(0);
19          echo("执行 remove(0) 后");
20          link.remove("Third");
21          echo("执行 remove(\"Third\") 后");
22          link.removeFirst();
23          echo("执行 removeFirst() 后");
24          link.removeFirstOccurrence("Fifth");
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



```
25          echo("执行 removeFirstOccurrence(\"Fifth\") 后");
26          link.removeLast();
27          echo("执行 removeLast() 后");
28          link.removeLastOccurrence("Fifth");
29          echo("执行 removeLastOccurrence(\"Fifth\") 后");
30      }
31      public static void echo(String action) {
32          System.out.println("");
33          System.out.println(action);
34          System.out.println("-------------------------");
35          for (int i = 0; i < link.size(); i++) {
36              System.out.println(i + " : " + link.get(i));
37          }
38          System.out.println("-------------------------");
39      }
40  }
```

【运行结果】运行此程序输出如下：

```
所有的元素
-------------------------
0 : First.
1 : Second
2 : Third
3 : Fourth
4 : Fifth
5 : Fifth
6 : Sixth
7 : Sixth
-------------------------
执行 remove() 后
-------------------------
0 : Second
1 : Third
2 : Fourth
3 : Fifth
4 : Fifth
5 : Sixth
6 : Sixth
-------------------------
执行 remove(0) 后
-------------------------
0 : Third
1 : Fourth
2 : Fifth
3 : Fifth
4 : Sixth
5 : Sixth
-------------------------

执行 remove("Third") 后
-------------------------
0 : Fourth
1 : Fifth
2 : Fifth
3 : Sixth
4 : Sixth
-------------------------
执行 removeFirst() 后
-------------------------
0 : Fifth
1 : Fifth
2 : Sixth
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
3 : Sixth
------------------------
执行 removeFirstOccurrence("Fifth") 后
------------------------
0 : Fifth
1 : Sixth
2 : Sixth
------------------------
执行 removeLast()后
------------------------
0 : Fifth
1 : Sixth
------------------------
执行 removeLastOccurrence("Fifth") 后
------------------------
0 : Sixth
------------------------
```

【代码解析】在以上程序中，首先实例化了一个 LinkedList 对象，在 main()方法中往链表添加了一系列元素（第 05～14 行），然后分别删除第一个和指定位置的元素，或是删除第一个出现或最后一个出现的元素。



22.4 栈和队列

栈和队列是两种特殊的线性表，各元素之间具有顺序的逻辑关系，可以采用顺序或链式的存储结构。与线性表不同的是，栈和队列的操作是受限的。

22.4.1 栈

栈常被称为 LIFO（Last In First Out）表，其元素的插入、取出和删除操作都是在一端进行，该端被称为“栈顶”，另一端称为“栈底”。由于栈的操作非常简单，其进栈、出栈、返回栈顶元素、判断是否空栈等操作都只消耗常数时间。

栈广泛地应用在程序设计中，比如表达式的处理，当程序运行到“(”时，将其及其后的元素都压入栈中；当遇到“)”时，依次将栈中元素弹出，计算其结果。

22.4.2 队列

队列常被称为 FIFO（First In First Out）表，其元素的插入被限制于一端，此端被称为“尾”，而元素的删除被限制于另一端，此端被称为“头”。队列的特点是，新的元素只能排在队尾，并且只能从队头取出元素。就像排队买饭一样，总是队头的人先买到。

队列的应用同样广泛，比如打印系统会将接到的多个打印任务插入到一个打印队列中，队列中的第一个任务会先打印，而最后一个任务会最后打印。

22.4.3 Java 实现栈

栈的主要功能是插入和删除，链表是最适合动态添加和删除的，所以下面用 LinkedList 实现一个栈。

【范例 22-5】演示链表实现的栈程序。

示例代码 22-5

```
01  // 栈接口，定义栈的基本功能
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



```
02  public interface Stack<T> {
03      boolean isEmpty();
04      void push(T t);
05      T peek();
06      T pop();
07  }
08  // 栈的实现类
09  public class LinkStack<T> implements Stack<T> {
10      private LinkedList<T> stack = new LinkedList<T>();
11      public boolean isEmpty() {
12          return stack.isEmpty();
13      }
14      public void push(T t) {
15          stack.addFirst(t);
16      }
17      public T peek() {
18          return stack.getFirst();
19      }
20      public T pop() {
21          return stack.removeFirst();
22      }
23      public static void main(String[] args) {
24          LinkStack<String> ls = new LinkStack<String>();
25          ls.push("apple");
26          ls.push("orange");
27          ls.push("banana");
28          System.out.println(ls.pop());
29          System.out.println(ls.peek());
30          System.out.println(ls.pop());
31          System.out.println(ls.isEmpty());
32      }
33  }
```

【运行结果】程序运行结果如下。

```
banana
orange
orange
false
```

【代码解析】在以上程序中，首先实现一个 Stack 接口，定义了栈的基本功能（第 01～07 行）。LinkStack 类实现了 Stack 接口，类中定义一个 LinkedList 变量用于存储栈的数据，然后分别使用 LinkedList 的相应函数实现 Stack 接口中定义的功能（第 11～22 行）。

22.4.4　Java 实现队列

同样用 LinkedList 实现一个队列。

【范例 22-6】示例代码 22-6 演示链表实现的栈程序。

示例代码 22-6

```
01  // 队列接口，定义队列的基本功能
02  public interface Queue<T> {
03      void insert(T t);
04      boolean isEmpty();
05      T remove();
06  }
07  // 队列的实现类
08  public class LinkQueue<T> implements Queue<T> {
09      private LinkedList<T> queue = new LinkedList<T>();
10      public void insert(T t) {
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
11          queue.addLast(t);
12      }
13      public boolean isEmpty() {
14          return queue.isEmpty();
15      }
16      public T remove() {
17          return queue.removeFirst();
18      }
19      public static void main(String[] args) {
20          LinkQueue<String> lk = new LinkQueue<String>();
21          lk.insert("Bentley");
22          lk.insert("Audi");
23          lk.insert("BMW");
24          System.out.println(lk.remove());
25          System.out.println(lk.remove());
26          System.out.println(lk.remove());
27          System.out.println(lk.isEmpty());
28      }
29  }
```

【运行结果】程序运行结果如下。

```
Bentley
Audi
BMW
True
```

【代码解析】在以上程序中，首先实现一个 Queue 接口，定义了队列的基本功能（第 01～06 行）。LinkQueue 类实现了 Queue 接口，类中定义一个 LinkedList 类型的私有变量用于存储队列的数据，然后使用 LinkedList 类的相应函数实现 Queue 接口的相应功能。

22.5 习题

【习题 22-1】已知一个数组中有若干字符串，写一个程序判断这些字符串是否有重复。

【分析】该题主要考查 Java 基本数据结构的应用。

【核心代码】

```
public class CheckDuplication {
    public static void main(String[] args) {
        String[] array = {"Jack", "Mike", "Rose", "Mike", "Peter"};
        List<String> ls = new ArrayList<String>();
        for (String s : array) {
            if (ls.contains(s)) {
                System.out.println("此数组存在重复");
                return;
            }
            ls.add(s);
        }
        System.out.println("此数组没有重复");
    }
}
```

【习题 22-2】写一个程序，利用递归算法打印出一个目录及其子目录下的所有文件的路径。

【分析】该题主要考查递归算法的应用。

【核心代码】

```
public class ListFile {
    public static void main(String[] args) {
        Scanner reader = new Scanner(System.in); // 扫描输入文本
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
        System.out.println("请输入文件路径：");
        String file = reader.next();
        list(new File(file));
    }

    public static void list(File d) {
        File[] fs = d.listFiles();
        for (File f : fs) {
            if (f.isDirectory()) {
                list(f);
            } else{
                System.out.println(f.getPath());
            }
        }
    }
}
```

【习题 22-3】写一个程序，使用递归算法计算 *n* 的阶乘。

【分析】该题主要考查递归算法的应用。

【核心代码】

```
public class Factorial {
    public static void main(String[] args) {
        Scanner reader = new Scanner(System.in);
        System.out.println("请输入数字");
        int n = reader.nextInt();

        System.out.println(n + "的阶乘：" + factorial(n));

    }

    public static BigDecimal factorial(int n) {
        if (n == 1) {
            return new BigDecimal(1);
        }
        return factorial(n - 1).multiply(new BigDecimal(n));
    }
}
```

【习题 22-4】写一个程序，使用递归算法计算两个数的最大公约数。

【分析】该题主要考查递归算法的应用。

【核心代码】

```
public class HighCommonDivisor {
    public static void main(String[] args) {
        Scanner reader = new Scanner(System.in);
        System.out.println("请输入两个数字，用空格隔开：");
        int x = reader.nextInt();
        int y = reader.nextInt();
        System.out.println(divisor(x, y));
    }

    public static int divisor(int x, int y) {
        if (x == y) {
            return x;
        } else {
            return divisor(Math.abs(x - y), Math.min(x, y));
        }
    }
}
```

【习题 22-5】写一个程序对一个字符串 List 进行排序。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

【分析】该题主要考查 Java 基本数据结构的应用。

【核心代码】

```
public class SortName {
    public static void main(String[] args) {
        List<String> ls = new ArrayList<String>();
        ls.add("Jerry");
        ls.add("Tom");
        ls.add("Rose");
        ls.add("Mike");
        ls.add("Harry");
        ls.add("Peter");
        System.out.println("未排序前：");
        for (String s : ls) {
            System.out.println(s);
        }

        Collections.sort(ls);
        System.out.println("排序后：");
        for (String s : ls) {
            System.out.println(s);
        }
    }
}
```

【习题 22-6】编写一个程序对包含 Student 类的 List 进行排序。

【分析】该题主要考查 Java 基本数据结构的应用。

【核心代码】

```
public class SortObject {
    public static void main(String[] args) {
        List<Student> ls = new ArrayList<Student>();
        ls.add(new Student("Mike", 70));
        ls.add(new Student("Harry", 89));
        ls.add(new Student("Tom", 76));
        ls.add(new Student("Marry", 83));
        ls.add(new Student("Rose", 65));
        ls.add(new Student("Peter", 83));
        System.out.println("排序前：");
        for (Student s : ls) {
            System.out.println(s.name + " "+ s.score);
        }
        Collections.sort(ls);
        System.out.println("排序后：");
        for (Student s : ls) {
            System.out.println(s.name + " "+ s.score);
        }
    }
    static class Student implements Comparable<Student> {
        private String name;
        private Integer score;
        public Student(String name, int score) {
            this.name = name;
            this.score = score;
        }
        public int compareTo(Student o) {
            if (null == o) {
                return 1;
            } else {
                return this.score.compareTo(o.score);
            }
        }
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



```
    }
}
```

【习题 22-7】写一个程序，使用栈对字符串反序输出。

【分析】该题主要考查 Java 基本数据结构的应用。

【核心代码】

```
public class ReverseString {
    public static void main(String[] args) {
        Scanner reader = new Scanner(System.in);
        System.out.println("请输入要反转的字符串：");
        String s = reader.next();
        System.out.println(reverse(s));
    }

    public static String reverse(String s) {
        Stack<String> stack = new Stack<String>();
        for (int i = 0; i < s.length(); i++) {
            stack.push(""+s.charAt(i));
        }
        StringBuffer sb = new StringBuffer();
        while (! stack.isEmpty()) {
            sb.append(stack.pop());
        }
        return sb.toString();
    }

    static class Stack<T> {
        private LinkedList<T> stack = new LinkedList<T>();

        public boolean isEmpty() {
            return stack.isEmpty();
        }
        public void push(T t) {
            stack.addFirst(t);
        }
        public T pop() {
            return stack.removeFirst();
        }
    }
}
```

【习题 22-8】利用栈写一个程序，判断一个表达式中的圆括号是否配对。

【分析】该题主要考查 Java 基本数据结构的应用。

【核心代码】

```
public class ValidateParentheses {
    public static void main(String[] args) {
        System.out.println(validate("(1+2)"));
    }
    public static String validate(String s) {
        Stack<String> stack = new Stack<String>();
        char c;
        int i=0;
        boolean isValid = true;
        String out;
        while (isValid && i < s.length()) {
            c = s.charAt(i);
            i++;
            switch (c) {
                case '(':
                    stack.push(c + "");
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
                break;
            case ')':
                out = stack.pop();
                if (out == null || !out.equals("(")) {
                    isValid = false;
                }

            }
        }
        if (isValid) {
            if (stack.isEmpty()) {
                return "ok";
            } else {
                return "need )";
            }
        } else {
            return "need (";
        }
    }

    static class Stack<T> {
        private LinkedList<T> stack = new LinkedList<T>();

        public boolean isEmpty() {
            return stack.isEmpty();
        }
        public void push(T t) {
            stack.addFirst(t);
        }

        public T pop() {
            if (isEmpty()) {
                return null;
            }
            return stack.removeFirst();
        }
    }
}
```

【习题 22-9】编写一个程序，将一个十进制的数字转换为八进制数，使用栈存储结果。

【分析】该题主要考查 Java 基本数据结构的应用。

【核心代码】

```
public class ConvertDecToOctal {
    public static void main(String[] args) {
        Scanner reader = new Scanner(System.in);
        System.out.println("请输入要反转的字符串：");

        int i =  reader.nextInt();
        Stack<String> stack = new Stack<String>();
        while (i > 0) {
            stack.push(i%8+"");
            i = i / 8;
        }
        while (!stack.isEmpty()) {
            System.out.print(stack.pop());
        }

    }

    static class Stack<T> {
        private LinkedList<T> stack = new LinkedList<T>();
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
        public boolean isEmpty() {
            return stack.isEmpty();
        }
        public void push(T t) {
            stack.addFirst(t);
        }

        public T pop() {
            if (isEmpty()) {
                return null;
            }
            return stack.removeFirst();
        }
    }
}
```

【习题 22-10】已知一个班有 m 个男生，n 个女生，m 与 n 不相等。要举行一个舞会，每曲开始时，依次从男生和女生中各出一人配对跳舞，本曲没有配对者则等待下一曲。输出每曲的配对情况。

【分析】该题主要考查 Java 基本数据结构的应用。

【核心代码】

```
public class DancerPair {
    public static void main(String[] args) {
        int m = 10; // 男同学的总数
        int n = 12; // 女同学的总数
        int x = 6, y = 5; // 判断这两个同学是否跳过舞
        Queue<Integer> bs = new Queue<Integer>();
        for (int i = 1; i <= m; i++) {
            bs.insert(i);
        }
        Queue<Integer> gs = new Queue<Integer>();
        for (int i = 1; i <= n; i++) {
            gs.insert(i);
        }
        int min = m > n ? n : m;
        int s =1;
        int c = 2;
        String result = "";
        while (s < 1000) {
            int b = bs.remove();
            int g = gs.remove();
            System.out.println("Boy"+b+ "<>" + "Girl"+g);
            bs.insert(b);
            gs.insert(g);

            if (b == x && g == y) {
                result += s;
                c--;
            }
            if (c == 0) {
                break;
            }
            s++;
        }
        if (c == 0) {
            System.out.println("Boy" + x + "and girl" + y + " dance together in: "
                               + result);
        } else {
            System.out.println("Boy"+x +" and Girl" + y + " have no chance to dance.");
        }
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
    }

    static class Queue<T> {
        private LinkedList<T> queue = new LinkedList<T>();

        public void insert(T t) {
            queue.addLast(t);
        }

        public boolean isEmpty() {
            return queue.isEmpty();
        }

        public T remove() {
            return queue.removeFirst();
        }
    }
}
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第 23 章　数　据　库

数据库技术如今发展十分迅速，已经广泛应用于各个行业，尤其在金融、电信等业务系统中占有十分重要的地位。作为一门强大的编程语言，Java 对数据库的支持也十分强大，通过 JDBC 工具可以方便地进行数据库编程。本章首先简单介绍数据库的知识，然后重点介绍 Java 中对数据库的支持。



23.1　数据库的介绍

数据库是根据某种数据模型组织起来存放到硬盘等二级存储设备中的数据集合。这种数据集合应该具有如下特点：尽可能不重复，以最优方式为某个组织的服务及其数据提供独立于应用程序的管理和控制，主要包括增、删、改和查找等操作。实际上，数据库系统是从文件系统进化而来的，它是一种更紧凑、更高级的文件系统。

数据库系统主要有如下特点。

- 数据独立性。数据的独立性是指在数据库中的数据是独立于应用程序的，也包括存储系统的物理结构与数据之间是相互独立的。
- 数据共享。数据共享是用户可以同时存取数据库中的数据，也包括用户可以通过各种方式来使用数据库。
- 减少数据冗余。同文件系统相比，数据库实现了数据的共享，从而使用户避免了重复建立相同的文件。这样不仅减少了数据冗余，而且便于维护数据的一致性。
- 数据集中控制。文件系统中数据处于一种松散的状态，用户之间、用户与文件之间以及文件与文件之间都是相互关联的。通过数据库模型来控制各种数据的组织。
- 数据一致性和可维护性。数据库中的数据具有数据一致性和可维护性，这样可以确保数据的安全性、完整性、并发控制以及故障的恢复。



23.2　MySQL 概述

MySQL 是当前应用最广泛的开源数据库软件，在非商业用途下可免费使用。MySQL 精小，但是其执行效率高，运行稳定，深受用户欢迎。无论是中小型商业应用还是学习之用，都是很不错的选择。

开发人员可以通过 MySQL 提供的 http://www.mysql.com/网站了解到有关 MySQL 的最新技术，并可以下载相关的软件。MySQL 网站的首页如图 23-1 所示。

23.2.1　MySQL 简介

MySQL 公司最初是由 David Axmark、Allan Larsson 和 Michael Monty Widenius 在瑞典创办的。MySQL 是一种开放源码软件，“开放源码”意味着任何人都能使用和改变软件，任何人都能从 Internet 下载 MySQL 软件，而无须支付任何费用。MySQL 软件采用了 GPL（GNU 通用公共许可证），如果开发人员需要在商业应用程序中嵌入 MySQL 代码，则可以从 MySQL 公

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

司购买商业许可版本。

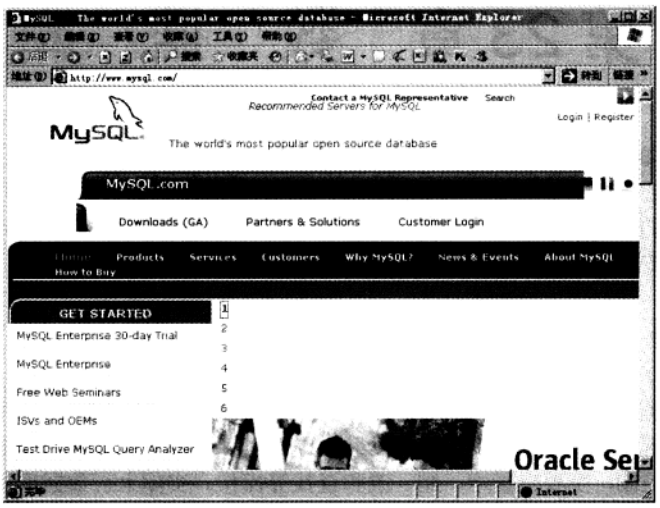


图 23-1　MySQL 网站的首页

23.2.2　下载并安装 MySQL

本书采用 MySQL 的 5.0 版本进行开发，因此，在 MySQL 网站首页的菜单栏上单击 Downloads 的链接，进入 MySQL 的下载界面，如图 23-2 所示。

在 MySQL 的下载画面提供了两种 MySQL 的下载方式，一种是供个人免费使用的版本，另一种是供商业应用的 MySQL 企业版。因为本书是做教材使用的，因此，下载 MySQL 的个人版可以直接单击 MySQL Community Server，即可进入 MySQL 个人版的下载界面，如图 23-3 所示。

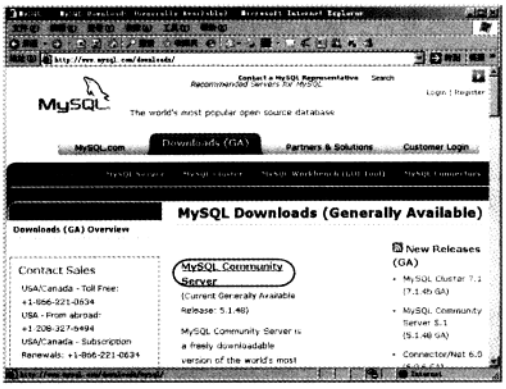


图 23-2　MySQL 下载界面

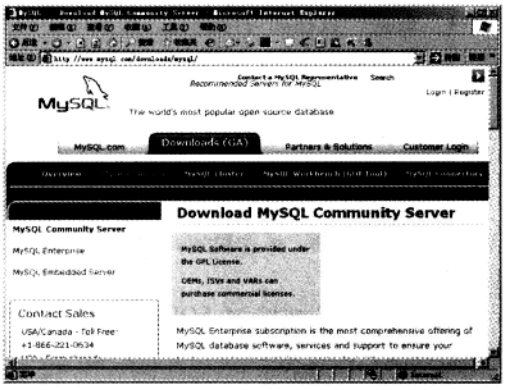


图 23-3　个人版下载界面

在 MySQL 个人版的下载画面提供了不同操作系统可以选择安装的版本，本书采用 Windows 32 位平台，因此，选择“Microsoft Windows”，同时显示符合条件的安装包，如图 23-4 所示。

在该页面中，用户可以选择是安装 MSI 文件还是解压缩文件，这里选择 MSI 文件安装方

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

式，单击 MSI 后面的“Download”按钮，即可进入下载的注册界面，如图 23-5 所示。

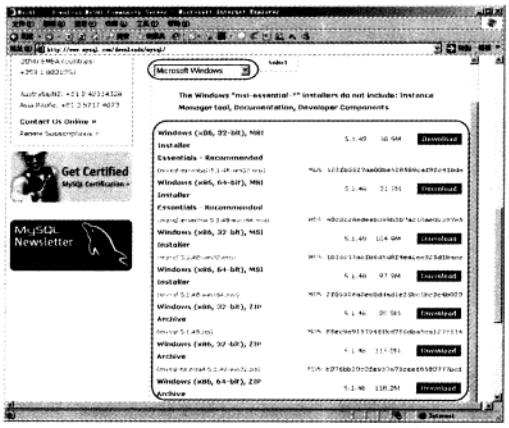


图 23-4　操作系统选择

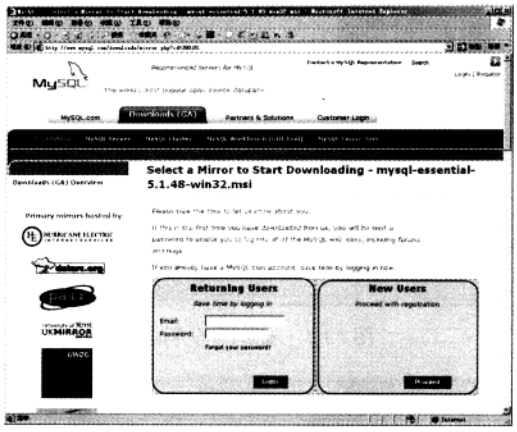


图 23-5　MySQL 注册界面

在下载前需要用户先登录，如果读者没有在 MySQL 的网站上注册过，则需要单击“New Users”中的“Proceed”按钮进行新用户注册；如果已经注册过，则直接在“Returning Users”中填入信息登录即可。进入登录后的下载界面如图 23-6 所示。

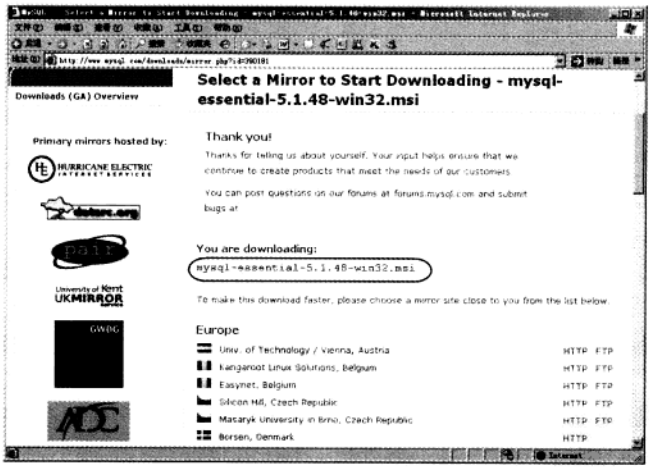


图 23-6　登录后的下载界面

进入登录成功的下载界面后，读者可以到不同的镜像网站进行下载，安装 MySQL 的步骤如下：

① 双击执行 MIS 文件，便可自动解压缩进行安装，首先出现的是安装程序准备页面，如图 23-7 所示。

② 单击“Next”按钮，即可进入选择安装类型界面，如图 23-8 所示。

③ 在选择安装类型界面中，选择单选框“Typical（典型）”，单击“Next”按钮，程序即可开始安装，安装界面如图 23-9 所示。

④ 安装结束后单击“Finish”按钮，则完成安装。

⑤ 接下来会弹出数据库安装界面，可以一直单击“Next”按钮，如图 23-10 所示。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

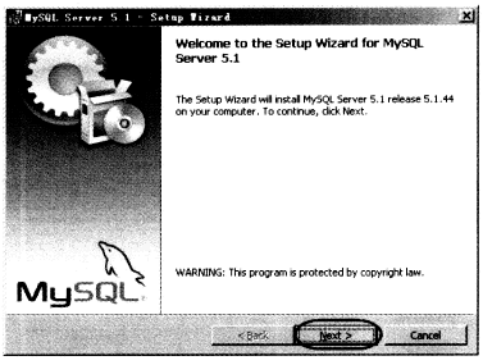


图 23-7　安装准备界面

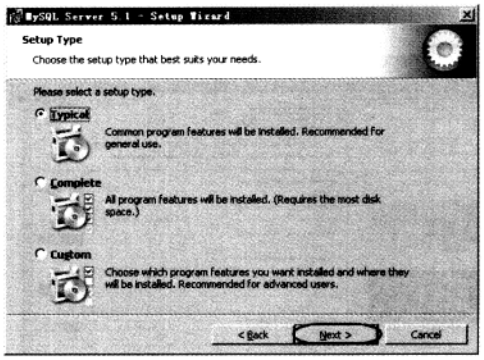


图 23-8　选择安装类型

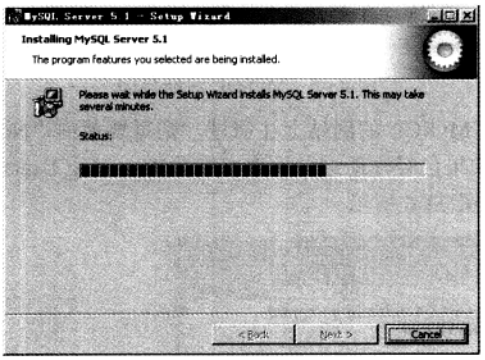


图 23-9　安装界面

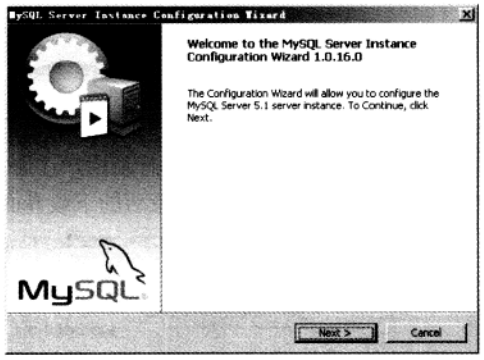


图 23-10　数据库安装

⑥ 在安装过程中，会提示用户选择数据库的编码格式，在此选择 gb3212，如图 23-11 所示，单击“Next”按钮。

⑦ 在弹出的密码设置界面需要设置数据库的访问密码，在此将密码设置为“123456”，如图 23-12 所示，单击“Next”按钮。

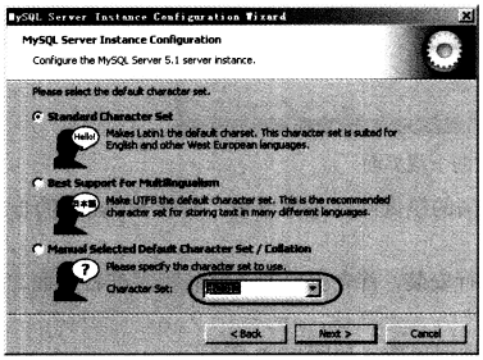


图 23-11　选择编码格式

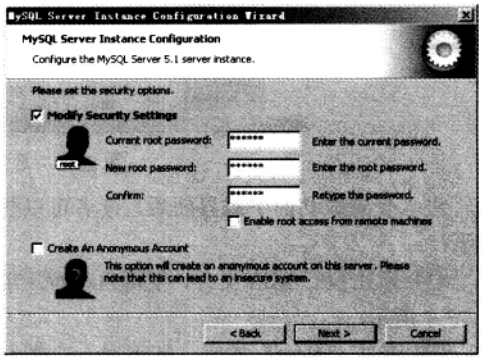


图 23-12　密码设置界面

⑧ 接下来开始进行数据库的安装，如果安装成功，则在分步安装提示的前面画“√”；如果安装失败，则在分步安装提示前画“×”，如图 23-13 所示。单击“Finish”按钮，完成安装。

安装完毕后，即可通过 Windows 的命令窗口来使用 MySQL，首先打开 Windows 的命令提

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

示符窗口，如图 23-14 所示。

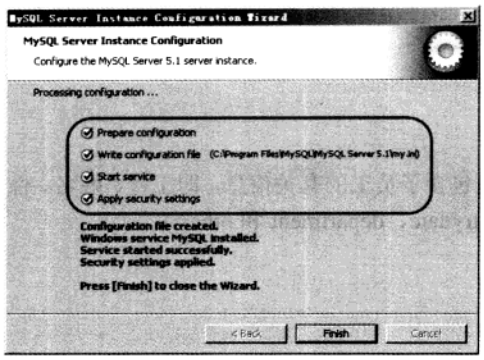


图 23-13　数据库安装成功

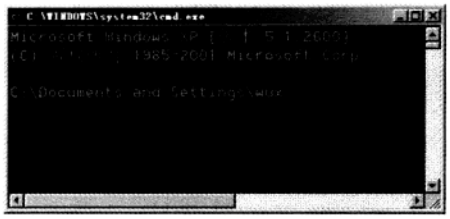


图 23-14　打开 Windows 的命令窗口

⑨ 进入 MySQL 的默认安装路径“C:\Program Files\MySQL\MySQL Server 5.1\bin”，如图 23-15 所示。



图 23-15　进入 MySQL 的默认安装路径

⑩ 输入命令“mysqld-nt”启动 MySQL，要验证是否安装成功，则可以通过输入命令“mysqlshow -u root”，查看是否能显示所有的数据库，如果显示，则表示安装成功。

⑪ 读者可以通过命令行界面进行数据库的各种操作。

23.2.3　MySQL 使用

配置好一个实例后，单击菜单“开始”→“所有程序”→“ MySQL”→“MySQL Server 5.0”→“MySQL Command Line Client”命令，打开 MySQL 的命令行窗口。按照提示输入配置时设置的密码，在 MySQL 命令行中输入上面设置的密码：root，进入命令行。



注意：MySQL 中的每条语句必须以“;”结束。

操作步骤如下：

① 查看数据库服务器上的数据库。

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| test               |
+--------------------+
3 rows in set (0.01 sec)
```

② 创建一个新的数据库 db。

```
mysql> CREATE DATABASE db;
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
Query OK, 1 row affected (0.03 sec)
```

③ 选择使用数据库。

```
mysql> use db;
```

④ 查看数据库内的表。

```
mysql> show tables;
Empty set (0.07 sec)
```

⑤ 创建一个数据库表。创建一个表，该表包含了员工的有关信息，即工号、姓名、性别、入职日期、部门和备注信息(id、name、sex、entrydate、department 和 others)。

```
mysql> CREATE TABLE Employees
    -> (
    -> id int not null,
    -> name varchar(10) not null,
    -> sex char(2) null,
    -> entrydate Date null,
    -> department varchar(20) null,
    -> others varchar(50) null
    -> )
    -> ;
Query OK, 0 rows affected (0.10 sec)
```

创建一个表后，再次使用 show tables 显示在数据库 db 中有哪些表。

```
mysql> show tables;
+--------------+
| Tables_in_db |
+--------------+
| employees    |
+--------------+
1 row in set (0.00 sec)
```

⑥ 显示表的结构。

```
mysql> describe employees;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| id         | int(11)     | NO   |     | NULL    |       |
| name       | varchar(10) | NO   |     | NULL    |       |
| sex        | char(2)     | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
3 rows in set (0.14 sec)
```

⑦ 给表添加新的记录。

```
mysql> insert into employees
    -> values('1','Jack','f','2008-3-4','Software Develpment','');
Query OK, 1 row affected (0.05 sec)
```

使用 SELECT 查看添加记录是否成功。

```
mysql> select * from employees;
+----+------+------+------------+--------------------+--------+
| id | name | sex  | entrydate  | department         | others |
+----+------+------+------------+--------------------+--------+
|  1 | Jack | f    | 2008-03-04 | Software Develpment |        |
+----+------+------+------------+--------------------+--------+
1 row in set (0.00 sec)
```

⑧ 文本数据的导入。

可以看出，按照上面的方式添加多条记录比较浪费时间，MySQL 提供了另外一种添加记录的方式：将记录按照一定的格式存放于文本文件，即每一个字段信息必须用定位符（Tab）

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

隔开。在 D:/Self-StudyManual/ch16/下创建一个文本文件“employees.txt”，每行包含一个记录，用定位符把值分开。使用下面的命令将文本文件“employees.txt”装载到 employees 表中。

```
mysql> LOAD DATA LOCAL INFILE "D:/Self-StudyManual/ch16/employees.txt" INTO TABLE
employees;
Query OK, 2 rows affected (0.05 sec)
Records: 2  Deleted: 0  Skipped: 0  Warnings: 0
```

查看是否添加成功。

```
mysql> select * from employees;
+----+------+------+------------+---------------------+-----------+
| id | name | sex  | entrydate  | department          | others    |
+----+------+------+------------+---------------------+-----------+
|  1 | Jack | f    | 2008-03-04 | Software Develpment |           |
|  2 | Lucy | m    | 2008-01-02 | Market              | nothing   |
|  3 | Lily | m    | 2007-02-05 | Production          | nothing   |
+----+------+------+------------+---------------------+-----------+
3 rows in set (0.00 sec)
```

⑨ 删除数据。

```
mysql> delete from employees where id = 3;
Query OK, 1 row affected (0.03 sec)
```

查看是否删除成功。

```
mysql> select * from employees;
+----+------+------+------------+---------------------+-----------+
| id | name | sex  | entrydate  | department          | others    |
+----+------+------+------------+---------------------+-----------+
|  1 | Jack | f    | 2008-03-04 | Software Develpment |           |
|  2 | Lucy | m    | 2008-01-02 | Market              | nothing   |
+----+------+------+------------+---------------------+-----------+
2 rows in set (0.00 sec)
```

⑩ 删除表。

```
mysql> drop table employees;
Query OK, 0 rows affected (0.04 sec)
```

查看表是否已删除。

```
mysql> show tables;
Empty set (0.05 sec)
```

⑪ 删除数据库。

```
mysql> drop database db;
Query OK, 0 rows affected (0.00 sec)
```

查看数据库是否已删除。

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| test               |
+--------------------+
3 rows in set (0.00 sec)
```

23.3　SQL 语句

SQL 语句用于执行数据库中的各种操作，例如，对数据的添加、删除、修改和更新等。在

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

一个数据库中可以包含有多个表，每个表都由行和列组成，并通过这些行和列来存储具体的信息，一行代表一条数据记录，例如，在用户信息中添加用户名、编号等。开发人员可以查询数据库表中所有的记录，也可以根据某一个列来查找相关的记录信息。如果查找信息，就需要遍历所有的记录，一个数据库表可以有上千条的记录，如果此时需要查询相关条件的信息，就需要对数据库进行优化，例如建立索引等，这样就可以大大提高查询的效率。SQL 语言主要由以下几部分组成。

- 数据定义语言(DDL，全称 Data Definition Language)。
- 数据操纵语言(DML，全称 Data Manipulation Language)。
- 数据控制语言(DCL，全称 Data Control Language)。
- 其他语言要素(ALE，全称 Additional Language Elements)。

在数据库中使用最多的就是数据库操纵语言，它包含了比较常用的 SQL 语句，其中包括：SELECT、INSERT、UPDATE、DELETE、CREATE、ALTER 和 DROP 七种基本操作。

23.3.1 查询语句 SELECT

SELECT 是查询语句，其作用是从表中查找符合某些条件的记录。它的使用规则如下：

```
SELECT <所选列的列名>
FROM 数据表名
[WHERE <条件表达式>]
[GROUP BY <列名 1> [HAVING <条件表达式>]]
[ORDER BY <列名 2> [ASC | DESC] ]
```

以上语句中，GROUP BY 子句用于查询结构汇总，并且要指明按照哪几个字段分组；ORDER BY 子句是按照一个或者多个字段来对查询结果进行排序，并且还可以选择对查询结果进行升序或降序排序，如果不指定，则默认为升序排列，而且该子句必须要放置到 SQL 语句的最后。

下面是一个查询数据表的示例，例如数据表的表名为 people，各列名为 id（编号）、name（姓名）、sex（性别）。使用 SELECT 语句查询女职员，并按照编号来进行排序，SQL 语句如下：

```
SELECT name
FROM people
WHERE sex='f'
ORDER BY id
```

23.3.2 插入语句 INSERT

INSERT 是插入语句，其作用是给已经存在的表中添加新的记录。它的使用规则如下：

```
INSERT INTO
"表名"[(属性 1,属性 2,...,属性 n)]
VALUES(属性 1,属性 2,...,属性 n)
```

下面是一个插入语句的示例，数据库表名为 people，其中各列分别为 id（编号）、name（姓名）、sex（性别）。使用 INSERT 语句添加一条新的语句，SQL 语句如下：

```
INSERT INTO
Employees [(id, name, sex)]
VALUES(00001, '张三', 'f')
```

23.3.3 更新语句 UPDATE

UPDATE 是更新语句，其作用是将数据表符合某个条件记录的某些字段更新。它的使用规

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



则如下：

```
UPDATE "数据表名"
SET<列名>=<表达式>
[WHERE<条件表达式>]
```

下面是一个使用 UPDATE 的示例，数据库表名为 people，数据库中各字段分别为 id（编号）、name（姓名）、sex（性别）。使用 UPDATE 语句更新 id 为 0001 的记录，并将 name 值修改为“李四”，SQL 语句如下：

```
UPDATE people
SET name='李四'
WHERE id=00001
```

23.3.4　删除语句 DELETE

DELETE 是删除语句，其作用是删除表中符合条件的一行或多行记录，它的使用规则如下：

```
DELETE FROM "数据库名" [WHERE<条件表达式>]
```

下面是一个使用 DELETE 语句删除一行记录的示例，假设数据表的名称为 people，数据表的各列分别为 id（编号）、name（姓名）、sex（性别），则可使用 DELETE 语句删除符合 id=0001 的记录，SQL 语句如下：

```
DELETE FROM people WHERE id=0001
```

23.3.5　建表语句 CREATE

CREATE Table 命令是创建一个数据库中不存在的新表，其使用规则如下：

```
CREATE TABLE 表名
({
列名 列类型
}[,...n]
)
```

以下示例是使用 CREATE 来创建一个表，该表包含了人的有关信息，如编号、姓名、性别 (id、name、sex)，SQL 语句如下：

```
CREATE TABLE people
(
id int not null,
name varchar(10) not null,
sex char(2) null
)
```

在上面所创建的表中，id 表示编号，类似于身份证号，而且是必须存在的，长度为 10，不能为空。name 表示姓名，数据类型为 varchar，长度为 10，不允许为空。sex 表示性别，数据类型为 char，长度为 2（因为该项只会输入“男”或“女”），允许为空。

23.3.6　修改表语句 ALTER

ALTER 语句用于对表增加列、删除列或者修改列，其使用规则如下：

```
ALTER TABLE 表名
{
ADD
{[]
|列名 列类型
|[]
}[,...n]
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
|DROP
{Column 列名称
}[,...n]
}
```



注意：在 ALTER 语句中可以通过 ADD 来添加列，使用 DROP 来删除列。有一些数据库不允许使用 DROP 删除表中的列。

如果需要修改表 Employees 的结构，即增加一个新字段，字段名为 city，数据类型是 varchar，长度是 23，SQL 语句如下：

```
ALTER TABLE Employees ADD city varchar(23)
```

23.3.7 删除表语句 DROP

DROP 语句是删除语句，其作用是删除已经存在的表。它的使用规则如下：

```
DROP TABLE 表名
```

例如，通过下面的程序删除已经创建的表 people，SQL 语句如下：

```
DROP TABLE people
```



23.4 JDBC 设计

JDBC 由 Java 编程语言编写的类和接口组成，同时它也为程序开发人员提供了一组用于实现对数据库访问的 JDBC API，用于执行 SQL 语句的接口。JDBC 隐藏了不同数据库之间的差异，实现了数据库无关的 Java 操作接口。这样，程序员就可以使用统一的 JDBC 接口，避免了直接处理操作底层数据库驱动程序与操作接口的差异。

JDBC API 是一种比较成熟的以面向对象编程的方式操作数据库的统一开放接口，它着重提供了一些基础的数据库调用级接口。

JDBC 和 Java 的结合体现了 Java“一次编写，到处运行”的特点。这样就使得程序员只写一次数据库程序，就可以在各种数据库系统上运行。程序员通过使用 JDBC 可以很方便地将 SQL 语句传给任何一种数据库，几乎只需要写同样的程序，就可以同时访问 MySQL 和 SQL Server。所以 JDBC 具有跨平台性。

23.4.1 JDBC 常见接口和类

java.sql 包提供了数据库编程需要的类和接口，利用它们可以方便地访问和处理数据库、具体的类、接口和异常。

1. JDBC 中比较常见的类

- Date：该类是接收处理数据库的 Date 类型数据。
- DriverManager：该类用于管理数据库驱动，完成注册、链接以及注销等。
- DriverPropertyInfo：该类用于保存处理数据库驱动的属性信息。
- Time：该类用于接收数据库的 Time 对象。
- Type：该类用于完成整数类别与各种数据类型一一对应。

2. JDBC 中比较常见的接口

- Array：该接口是 Java 与 SQL 的 ARRAY 类型的映射。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



- Blob：该接口是 Java 与 SQL 的 Blob 类型的映射。
- CallableStatemet：该接口用于执行 SQL 存储语句。
- Clob：该接口是 Java 与 SQL 的 Clob 类型的映射。
- Connection：该接口用于表示数据库的连接。
- DatabaseMetaData：该接口用于表示数据库相关的信息。
- Driver：该接口是驱动程序必须实现的接口。
- ParameterMetaData：该接口用于获取 PreparedStatement 对象中每一个参数标记的类型和属性的对象。
- PreparedStatement：该接口是 SQL 语句被编译并存储于 PreparedStatement 的对象。
- Ref：该接口是 Java 与 SQL 中的 REF 值的映射，它是到数据库 SQL 结构类型值的引用。
- ResultSet：该接口用于表示接收 SQL 语句并返回数据库结果集的数据表。
- ResultSetMetaData：该接口用于获取数据库返回的结果集的列的类型和属性信息。
- SQLData：该接口是 Java 与用户自定义类型的映射。
- Statement：该接口用于执行静态 SQL 语句，并返回结果。
- Struct：该接口是 Java 与 SQL 的 Structured 类型的映射。

3. JDBC 中比较常见的异常处理

- BatchUpdatedExceptions：该异常用于进行批处理的更新操作时抛出异常。
- DataTruncation：该异常用于数据被意外截断抛出异常。
- SQLException：该异常用于数据库访问错误或者其他错误信息。
- SQLWarning：该异常用于提供关于数据库访问警告信息异常。

23.4.2　JDBC 数据库连接步骤

使用 JDBC 连接数据库，首先要加载 JDBC 驱动程序，接下来要识别加载的驱动程序，然后进行数据库的操作，并将结果放置到结果集中，对结果集进行遍历，最后要关闭数据库的连接。使用 JDBC 连接数据库的步骤如下。

① 使用 Class.forName()方法来加载程序中的 JDBC 驱动程序。其中需要注意的就是要确定设置好 Class 路径，确保 JDBC 在驱动路径中。例如，如下语句：

```
Class.forName("org.gjt.mm.mysql.Driver");
```

② 在程序加载完成 JDBC 驱动之后，DriverManager 就会去识别所加载的驱动程序，并调用 getConnection()方法来连接数据库。DriverManager 是用于管理 JDBC 驱动程序的类，例如，如下与 MySQL 数据库建立连接的方法：

```
String url="jdbc:mysql://localhost/MySqlDB";                    // 其中 MySqlDB 为数据库名
String user="root";
String password="admin";
Connection conn=DriverManager.getConnection(url,user,password);
```

③ 使用 getConnection()方法获取 Connection 对象之后，就可以创建一个 Statement 对象对数据库进行操作。例如，如下语句：

```
Statement stmt=conn.createStatement();
//创建一个 statement 的实例，其中 conn 是 Connection 的实例。
```

④ 接下来就需要返回一个结果集，通过执行 Statement 对象的 executeQuery()方法来返回 Resultset 结果集对象，最后对结果集进行遍历。例如，如下遍历结果集的语句：

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
String query="select * from people";
ResuletSet rs=stmt.executeQuery(query);
while(rs.next())
{
...
}
```

⑤ 程序的最后，也就是使用完数据库之后，要关闭数据库的连接，其中包括前面所使用的 Resuletset 对象、Statement 对象和 Connection 对象的关闭。

```
rs.close();                    //关闭 Resulet 对象
stmt.close();                  //关闭 Statement 对象
conn.close();                  //关闭 Connection 对象
```

23.4.3 加载 JDBC 驱动程序

加载 JDBC 驱动程序可以使用 Class.forName 方法，并且要向 DriverManager 注册 JDBC 驱动程序。MySQL 的驱动程序类是 com.mysql.jdbc.Driver 或 org.gjt.mm.mysql.Driver，连接 MySQL 的应用语法如下：

```
Class.forName("com.mysql.jdbc.Driver");
```

或者如下：

```
Class.forName("org.gjt.mm.mysql.Driver");
```

DriverManager 类的主要方法包括以下几种。

- 从驱动器列表中删除一个驱动程序：static void deregisterDriver(Driver driver)。
- 建立给定数据库的连接：static Connection getConnection(String url)、static Connection getConnection(String url, Properties info)、static Connection getConnection(String url, String user, String password)。
- 查找给定 URL 的驱动程序：static Driver getDriver(String url)。



注意：连接数据库之前，需要加载数据库的驱动，MySQL 的数据库驱动就是一个 jar 文件（mysql-connector-java-5.1.6-bin.jar），可以从 MySQL 的官方网站中进行下载。如果使用 javac 命令编译，需要将下载到的 jar 文件放置到一个文件中，例如，C:\Java\jdk1.6.0_10\lib\ext\目录下，然后需要在环境变量 classpath 中配置该文件路径，例如 C:\Java\jdk1.6.0_10\lib\ext\mysql-connector-java-5.1.6-bin.jar。如果使用 MyEclipse 等开发工具，则可以直接将 jar 文件放置到 lib 目录下。

下面是一个加载 JDBC 驱动的程序片段。

```
01  public class DriverManagerMySQLTest {
02      public static void main(String[] args) {
03          try {
04          lass.forName("com.mysql.jdbc.Driver");      //动态加载和创建 Class 对象
05              System.out.println("找到驱动，连接成功…");
06          }
07          catch(ClassNotFoundException e) {           //捕获异常
08              System.out.println("找不到驱动程序类");   //输出字符串信息
09          }
10      }
11  }
```

程序运行结果如图 23-16 所示。

在以上程序中，首先要正确配置数据库驱动，然后设置数据库驱动（第 04 行），设置连接

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

数据库，并在连接成功后输出“找到驱动，连接成功...”字样（第 05 行）。最后设置一个异常，如果没有找到驱动，则输出“找不到驱动程序类”（第 07、08 行）。

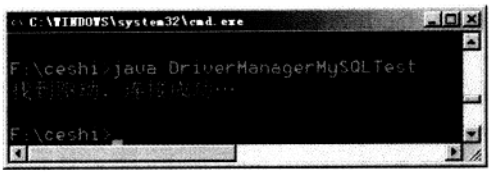


图 23-16　DriverManagerMySQLTest.java 运行结果

23.4.4　获取 Connection 连接对象

Connection 对象代表和数据库的连接，连接的过程包括所有执行的 SQL 语句和在该连接上所返回的结果。一个应用程序中可以和单个数据库有一个或多个连接，也可以与许多数据库有连接。使用适当的驱动程序与数据库管理系统建立一个连接就是打开数据库连接。建立连接的方法是 DriverManager 加载驱动程序 Driver 类之后，就可以与数据库建立联系了。

连接对象表示与特定数据库建立的连接(会话)。一个应用程序可以与特定的数据库建立一个或多个连接，并且可以和多个数据库建立多个连接。Connection 类的主要方法如下。

- void close()：该方法的作用是关闭连接，释放占用的资源。
- void commit()：该方法的作用是使上次提交变为永久更改，并释放 Connection 的所有数据锁。
- Statement createStatement()：该方法的作用是创建一个 Statement 对象。
- Statement createStatement(int resultSetType, int resultSetConcurrency)：该方法的作用是创建一个指定类型和并发性的 Statement 对象。
- Statement createStatement(int resultSetType, int resultSetConcurrency, int resultSetHoldability)：该方法的作用是创建一个指定类型、并发性和可保存性的 Statement 对象。
- boolean isClosed()：该方法的作用是查询 Connection 对象是否关闭。
- PreparedStatement prepareStatement(String sql)：该方法的作用是创建一个 PreparedStatement 对象。
- PreparedStatement prepareStatement(String sql, int autoGeneratedKeys)：该方法的作用是创建一个能自动生成键的 PreparedStatement 对象。
- PreparedStatement prepareStatement(String sql, int[] columnIndexes)：该方法的作用是创建一个由给定数组产生的 PreparedStatement 对象。
- PreparedStatement prepareStatement(String sql, int resultSetType, int resultSetConcurrency)：该方法的作用是创建一个指定类型和并发性的 PreparedStatement 对象。
- PreparedStatement prepareStatement(String sql, int resultSetType, int resultSetConcurrency, int resultSetHoldability)：该方法的作用是创建一个指定类型、并发性和可保存性的 PreparedStatement 对象。
- PreparedStatement prepareStatement(String sql, String[] columnNames)：该方法的作用是创建一个由给定数组指定的 PreparedStatement 对象。
- void rollback()：该方法的作用是取消事务过程中的所有更改，并释放该对象持有的数据库锁。
- void rollback(Savepoint savepoint)：该方法的作用是取消创建 savepoint 之后进行的更改。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

JDBC URL 定义了连接数据库时的协议、子协议、数据源识别，一般格式如下：

```
Connection con=DriverManager.getConnection("协议:子协议:数据源识别");
```

“协议”在 JDBC 中总是以 jdbc 开始的，“子协议”是驱动程序名称或数据库管理系统名称，使用 MySQL 数据库，该子协议就是“mysql”；“数据源识别”标识数据源连接的 URL 地址与连接端口。MySQL 的 JDBC URL 书写格式如下：

```
jdbc:mysql://主机地址:连接端口/数据库名称?参数=值&参数=值
```

主机名称可以是本机或其他主机的地址，默认连接端口为 3306，若要连接 DB 数据库，用户名称与密码都是“root”，JDBC URL 可以写为：

```
jdbc:mysql://localhost:3306/db?user=root&password=123
```



注意：如果数据库中的数据包括中文。为了防止出现乱码现象，还必须给定参数 useUnicode 及 characterEncoding，它们分别表示是否使用 Unicode 和字符编码方式，例如：

```
jdbc:mysql://localhost:3306/demo?user=root&password=123&useUnicode=true&characterEncoding=UTF-8
```



23.5 Java 与各种数据库连接

JDBC 与对各种数据库的操作都使用统一的 SQL 语句，因此，数据库的各种操作基本上是相同的。不同之处在于数据库的连接上，不同的数据库，必须使用各自的驱动程序(驱动程序可以在 Sun 公司的网站上下载)。

1. 连接 Oracle 数据库

JDBC 连接 Oracle 数据库代码如下：

```
//载入驱动程序类别
Class.forName("oracle.jdbc.driver.OracleDriver").newInstance();
//创建，并初始化数据库 url
String url="jdbc:oracle:thin:@localhost:1521:orcl";
//数据库用户名
String user=" admin ";
//数据库密码
String password=" 123456";
//获得 Connection 对象
Connection dbConn= DriverManager.getConnection(url,user,password);
```

2. 连接 DB2 数据库

JDBC 连接 DB2 数据库的代码如下：

```
//载入驱动程序类别
Class.forName("com.ibm.db2.jdbc.app.DB2Driver").newInstance();
//创建，并初始化数据库 url
String url="jdbc:db2://localhost:5000/dbname";
//数据库用户名
String user=" admin ";
//数据库密码
String password="123456";
//获得 Connection 对象
Connection dbConn = DriverManager.getConnection(url,user,password);
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

3. 连接 SQL Server 数据库

JDBC 连接 SQL Server 数据库的代码如下：

```
//载入驱动程序类别
Class.forName("com.microsoft.jdbc.sqlserver.SQLServerDriver").newInstance();
//创建，并初始化数据库 url
String url="jdbc:microsoft:sqlserver://localhost:1433;DatabaseName=dbname";
//数据库用户名
String user=" admin ";
//数据库密码
String password="123456";
//获得 Connection 对象
Connection dbConn = DriverManager.getConnection(url,user,password);
```

4. 连接 Sybase 数据库

JDBC 连接 Sybase 数据库的代码如下：

```
//载入驱动程序类别
Class.forName("com.sybase.jdbc.SybDriver").newInstance();
//创建，并初始化数据库 url
String url =" jdbc:sybase:Tds:localhost:5007/dbname";
//获取系统属性
Properties sysProps = System.getProperties();
//数据库用户名
SysProps.put("user","admin");
//数据库密码
SysProps.put("password","123456");
//获得 Connection 对象
Connection dbConn = DriverManager.getConnection(url, SysProps);
```

5. 连接 Access 数据库

JDBC 连接 Access 数据库的代码如下：

```
//载入驱动程序类别
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
//获得 Connection 对象
Connection dbConn = DriverManager.getConnection("jdbc:odbc:dbname");
//dbname 为数据源的名称
```



23.6　习题

【习题 23-1】对任意数据库进行操作之前，都需要先进行连接，而连接的过程中必须加载 JDBC 驱动程序。编写一个加载 JDBC 驱动的类。

【分析】本题主要考查加载 JDBC 的驱动类。

【核心代码】

```
try {
    class.forName("com.mysql.jdbc.Driver");      //动态加载和创建 class 对象
       System.out.println("找到驱动，连接成功…");
    }
    catch(ClassNotFoundException e) {            //捕获异常
       System.out.println("找不到驱动程序类");     //输出字符串信息
    }
```

【习题 23-2】编写一个程序，实现对数据的更新，其中包括表的创建、删除和记录属性的增删以及记录的增、删、改操作。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

【分析】本题主要考查对 SQL 语句的使用。

【核心代码】

```
CREATE TABLE people
(
id int not null,
name varchar(10) not null,
sex char(2) null
)
DELETE FROM people WHERE id=0001
```

【习题 23-3】编写一个程序，连接 Oracle 数据库，并返回 Connection 对象。

【分析】本题主要考查 JDBC 连接数据库。

【核心代码】

```
public class ConnOracle {
    public static void main(String[] args) throws ClassNotFoundException,
IllegalAccessException, InstantiationException, SQLException {
        Class.forName("oracle.jdbc.driver.OracleDriver").newInstance();
        Connection con = DriverManager.getConnection("jdbc:oracle:thin:@local-
host: 1521:db", "root", "root");
    }
}
```

【习题 23-4】对于频繁访问数据库的程序来说，选择批处理是最合适不过的了。编写一个程序，实现批处理。也就是一次执行多条 SQL 语句，这些 SQL 语句可以是插入语句、更新语句和删除语句。

【分析】本题主要考查对数据库批处理的使用。

【核心代码】

```
public class BatchSQL {
   public static void main(String[] args) throws ClassNotFoundException,
SQLException {
      Class.forName("com.mysql.jdbc.Driver");
      String uri = "jdbc:mysql://localhost:3306/mydb";
      String user = "root";
      String passwd = "root";
      Connection conn = DriverManager.getConnection(uri, user, passwd);
      Statement st = conn.createStatement();
      conn.setAutoCommit(false);
      for (String s : args) {
         st.addBatch(s);
      }
      st.executeBatch();
      conn.commit();
      conn.setAutoCommit(true);
   }
}
```

【习题 23-5】编写一个工具类，负责连接数据库，创建 Connection 对象。

【分析】本题主要考查 Connection 的创建。

【核心代码】

```
public class JDBCTool {
   public static Connection getConnection(String url, String user, String passwd)
throws ClassNotFoundException, SQLException {
      Class.forName("com.mysql.jdbc.Driver");
      return DriverManager.getConnection(url, user, passwd);
   }
}
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



【习题 23-6】编写一个创建 Article 数据表的 SQL 语句。

【分析】本题主要考查数据表的创建。

【核心代码】

```
create table ARTICLE(
   id int not null primary key,
   title varchar(20) not null,
   content text not null
);
```

【习题 23-7】编写一个程序按给定的条件查找相应的记录。

【分析】本题主要考查 PreparedStatement 的使用。

【核心代码】

```
public class GetUser {
   public static void main(String[] args) throws ClassNotFoundException,
SQLException {
      Connection con = JDBCTool.getConnection("jdbc:mysql://localhost:3306/mydb",
                                              "root", "root");
      PreparedStatement ps = con.prepareStatement("select * from USER where name=?
                                                   and passwd=?");
      ps.setString(1, "Mike");
      ps.setString(2, "passwd");
      ResultSet rs = ps.executeQuery();
      while (rs.next()) {
         System.out.println(rs.getString("name") + " : " + rs.getString("passwd"));
      }
      rs.close();
      ps.close();
      con.close();
   }
}
```

【习题 23-8】编写一个程序，取出所有的 Article，并打印 title。

【分析】本题主要考查 ResultSet 的遍历。

【核心代码】

```
public class ListResult {
   public static void main(String[] args) throws ClassNotFoundException,
SQLException {
      Connection con = JDBCTool.getConnection("jdbc:mysql://localhost:3306/mydb",
                                              "root", "root");
      Statement st = con.createStatement();
      ResultSet rs = st.executeQuery("select * from ARTICLE");
      while (rs.next()) {
         String title = rs.getString("title");
         String content = rs.getString("content");
         System.out.println(title);
      }
      rs.close();
      st.close();
      con.close();
   }
}
```

【习题 23-9】编写一个程序，使用 PreparedStatement 向数据库插入一条记录。

【分析】本题主要考查 PreparedStatement 的使用。

【核心代码】

```
public class PrepareStatement {
   public static void main(String[] args) throws ClassNotFoundException,
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
SQLException {
            Connection con = JDBCTool.getConnection("jdbc:mysql://localhost:3306/mydb",
                                                    "root", "root");
            PreparedStatement ps = con.prepareStatement("insert into USER value(?, ?)");
            ps.setString(1, "Mike");
            ps.setString(2, "passwd");
            ps.executeUpdate();
            ps.close();
            con.close();
        }
    }
```

【习题 23-10】编写一个程序，向数据库中插入一条记录，然后取出，并打印出内容。

【分析】本题主要考查 JDBC 插入数据，并查询数据。

【核心代码】

```
    public class ListUser {
        public static void main(String[] args) throws ClassNotFoundException,
SQLException {
            Connection con = JDBCTool.getConnection("jdbc:mysql://localhost:3306/mydb",
                                                    "root", "root");
            Statement st = con.createStatement();
            st.executeUpdate("insert into USER value('Jack', 'passwd");
            ResultSet rs = st.executeQuery("select * from USER");
            while (rs.next()) {
                System.out.println("名称：" + rs.getString("name") + " 密码：" +
                                   rs.getString("passwd"));
            }
        }
    }
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第 24 章　Applet 编程

Applet 是一些应用小程序，使用它可以在浏览器的环境中运行，例如，打开一个网页时，出现的一些动态文字显示或者是一些绚丽的显示效果。在 Applet 中并没有普通程序中的 main。本章就介绍一下 Applet 程序开发。



24.1　Applet 类

在 Java 中主要分为 Applet 程序和应用程序两种，前面介绍的都是应用程序，二者的区别就在于应用程序中都包含有一个 main 方法；Applet 程序中没有 main 方法。Applet 嵌套在网页中，可以使页面显得更加丰富多彩。

24.1.1　Applet 简介

Applet 程序可以运行于“Applet Viewer”或者是 Web 浏览器中，它的扩展名和所有的 Java 程序相同，也是“*.Java”，被编译过的 Applet 程序扩展名为“*.class”。

Applet 小程序可以嵌入到页面中，由支持它的浏览器来解释执行，包含有 Java Applet 的网页被称为 Java-Powered，也可以称为 Java 支持的网页。使用 Java Applet 的网页可以产生一些绚丽的效果，并且能够提高 Web 的交互能力和动态执行能力。

当用户访问包含有 Applet 的网页时，客户端就会下载到 Applet 字节码文件，只有这样，客户端才会去解释和执行 Applet 程序，这样就暴露了一个安全问题。如果 Applet 中包含有恶意代码，就会对客户端带来损失，实际上，多数的浏览器都对 Applet 程序做了一些限定，用于保护客户端。

- 禁止运行本地可执行文件。
- 禁止与除服务器之外的其他主机通信。
- 禁止读写本地文件。
- 不能装载动态链接库。
- 不能调用本地方法。
- 禁止访问用户名、密码、电子邮件地址以及本地计算机信息。

24.1.2　Applet 的开发流程

Applet 也具有 Java 的跨平台性，不同平台可以运行相同的代码。用户在每次打开包含有 Applet 的页面时，Applet 就会被自动下载。Applet 的基本流程如下。

① 浏览器打开网页时，如果网页中包含有 Applet 程序，浏览器就会自动开启 Java 虚拟机并运行。

② 浏览器将<Applet>标记给定的信息传送给虚拟机的类加载器。

③ 在类加载时，就会根据一个认证机制来检查该类是否合法，然后通过检查的结果来判断是否执行 Applet 程序。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

24.1.3 Applet 的方法

Applet 类是开发 Applet 程序的基础，所有的 Applet 程序都必须要继承 Applet 类，它提供了一些标准的接口，比较常用的如下。

- void destroy()：由浏览器或 Applet viewer 调用，通知此 Applet，它正在被回收，它应该销毁分配给它的任何资源。
- AccessibleContext getAccessibleContext()：该方法的作用是获取与此 Applet 关联的 AccessibleContext。
- AppletContext getAppletContext()：该方法的作用是确定此 Applet 的上下文，上下文允许 Applet 查询和影响它所运行的环境。
- String getAppletInfo()：该方法的作用是返回有关此 Applet 的信息。
- AudioClip getAudioClip(URL url) ：该方法的作用是返回 URL 参数指定的 AudioClip 对象。
- AudioClip getAudioClip(URL url, String name) ：该方法的作用是返回 URL 和 name 参数指定的 AudioClip 对象。
- URL getCodeBase()：该方法的作用是返回当前 Applet 的基地址 URL。
- URL getDocumentBase()：该方法的作用是获取嵌入此 Applet 的文档的 URL。
- Image getImage(URL url) ：该方法的作用是返回能被绘制到屏幕上的 Image 对象。
- Image getImage(URL url, String name) ：该方法的作用是返回能被绘制到屏幕上的 Image 对象。
- Locale getLocale()：该方法的作用是获取 Applet 的语言环境。
- String getParameter(String name) ：该方法的作用是返回 HTML 标记中指定参数的值。
- String[][] getParameterInfo()：该方法的作用是返回此 Applet 理解的关于参数的信息。
- void init()：该方法的作用是由浏览器或 Applet viewer 调用，通知此 Applet，它已经被加载到系统中。
- boolean isActive()：该方法的作用是确定 Applet 是否处于活动状态。
- static AudioClip newAudioClip(URL url) ：该方法的作用是从给定 URL 处获取音频剪辑。
- void play(URL url) ：该方法的作用是播放指定绝对 URL 处的音频剪辑。
- void play(URL url, String name) ：该方法的作用是播放音频剪辑，给定了 URL 及与之相对的说明符。
- void resize(Dimension d) ：该方法的作用是请求调整此 Applet 的大小。
- void resize(int width, int height) ：该方法的作用是请求调整此 Applet 的大小。
- void setStub(AppletStub stub) ：该方法的作用是设置此 Applet 的 stub，系统自动调用。
- void showStatus(String msg) ：该方法的作用是请求将参数字符串显示在“状态窗口”中。
- void start()：该方法的作用是由浏览器或 Applet viewer 调用，通知此 Applet 应该开始执行。
- void stop()：该方法的作用是由浏览器或 Applet viewer 调用，通知此 Applet 应该终止执行。

24.1.4 Applet 的生命周期

Applet 程序中的生命周期与应用程序相比，就显得比较简单了，Applet 中共有四个状态，

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



分别是初始态、运行态、停止态和消亡态，相对应 Applet 的四个方法分别是 init()、start()、stop()、destroy()。

在调用 init()方法之后，Applet 就会进入初始状态；然后就要调用 start()方法，Applet 就会进入运行状态；在窗口最小化的时候，就会调用 stop()方法，此时的 Applet 就会停止，进入停止状态；此时如果再次还原回窗口时，会再次调用 start()方法；如果关闭窗口或者关闭浏览器窗口时，就会调用 destroy()方法进入消亡状态。

1．init()方法

该方法是在 Applet 程序第一次被运行时进行加载执行。在 Applet 程序的生命周期中只对该方法执行一次，该方法的内部主要是做一些程序的初始化工作，例如，获取参数、为组件添加事件监听等，在该方法中还可以添加线程。

2．start()方法

start()方法是 Applet 的主体方法，它可以运行重复执行的任务，或者是激活一个线程任务。

3．stop()方法

stop()方法和 start()方法是相反的，stop()是停止程序的运行，当浏览器最小化的时候，浏览器就会自动调用 stop()方法，其目的主要是为了节约资源。该方法在一个完整的生命周期中会被调用多次。

4．destroy()方法

destroy()方法和 init()方法相同，在一次完整的生命周期中只能够执行一次，就是当浏览器或者是窗口关闭的时候，调用该方法，释放资源。前面提到在 init()方法中创建线程，那么就可以在 destroy()方法中结束线程。

【范例 24-1】示例代码 24-1 是一个使用字体变大示例。

示例代码 24-1

```
01  public class WordsTest extends Applet implements Runnable {
02      private Image image;                          // 定义图片对象
03      private Graphics graphic;                     // 定义 Graphics 对象
04      private Font font;                            // 定义 Font 对象
05      private String str;                           // 定义一个字符串对象
06      private Thread thread;                        // 定义一个 Thread 对象
07      private int fontSize;                         // 定义一个 int 型变量
08      public void init() {                          // Applet 小程序初始化
09          this.setSize(400, 400);
10          image = createImage(400, 400);            // /创建 Image 图像对象
11          graphic = image.getGraphics();
12          str = "这是 Applet 小程序";                 // 需要变化的文本内容
13          font = new Font("TimesRoman", Font.BOLD, 8);
14      }
15      public void start() {                         // 启动多线程
16          if (thread == null) {
17              thread = new Thread(this);
18              thread.start();                       // 启动线程
19          }
20      }
21      public void update(Graphics g) {              // 重新调用 paint 方法
22          paint(g);
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
23        }
24        public void paint(Graphics g) {                    // 绘图像
25            graphic.setColor(Color.black);
26            graphic.fillRect(0, 0, getSize().width, getSize().height);
27            font = new Font("TimesRoman", Font.BOLD, fontSize);
28            graphic.setFont(font);
29            graphic.setColor(Color.white);
30            FontMetrics fm = graphic.getFontMetrics(font);
31            int fontHeight = fm.getHeight();
32            int w;
33            int baseLine = getSize().height / 2 + fontHeight / 2;
34            w = fm.stringWidth(str);
35            w = (getSize().width - w) / 2;
36            graphic.drawString(str, w, baseLine -= 20);
37            g.drawImage(image, 0, 0, this);
38            fontSize++;
39        }
40        public void run() {                                // 重写 run 方法
41            while (true) {
42                repaint();
43                if (fontSize > getSize().height)
44                    fontSize = 0;
45                try {
46                    thread.sleep(100);
47                } catch (InterruptedException e) {
48                }
49            }
50        }
51    }
```

【运行结果】运行结果如图 24-1 和图 24-2 所示。

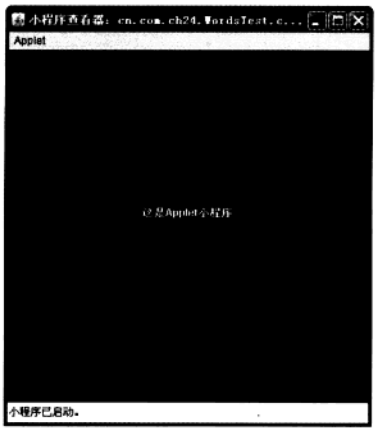


图 24-1　Applet 小程序初始化



图 24-2　字体变大

【代码解析】以上程序中，WordsTest 类继承于 Applet，并实现 Runnable 接口，在 init()方法中对 Applet 程序进行初始化（第 08～14 行）。start()方法中为开启线程（第 15～20 行），run()方法中设置线程的休眠时间为 100ms（第 40～48 行）。

24.2　Applet 应用

Applet 中还可以显示动态的图形，播放一些多媒体文件等。本节就主要介绍 Applet 中的应

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

用，并通过示例来实现这些功能。

24.2.1　Applet 中的文字处理

在 Applet 中可以对文字实现比较多的效果，例如，像灯光一样在文字上照射且字体慢慢放大的效果灯，本节就通过几个示例来演示对文字的处理。

1．模拟灯光照射文字效果

使用 Applet 和多线程来完成一个模拟灯光的效果，也就是为文字设置一个底色，然后用白色模拟灯光，在文字上照射，被照到的部分会变成白色。

【范例 24-2】示例代码 24-2 是一个灯光艺术字示例。

示例代码 24-2

```
01  public class LightFont extends Applet implements Runnable {
02      private String text;                        // 声明文本对象
03      private Font font;                          // 声明字体对象
04      private int fontSize;                       // 声明字体的大小
05      private Thread thread;                      // 声明线程对象
06      private int position = 100;                 // 声明光标点出现的位置
07      private int tsize = 20;                     // 文本的大小
08      private int twidth = 0;                     // 文本的宽度
09      private int fheigth, baseLine, w;
10      public void init()                          // 小程序的初始化
11      {
12          text = "灯光艺术字";                    // 在小程序中需要显示的文本对象
13          fontSize = 30;                          // 设置字体的大小
14          font = new Font("TimesRoman", Font.BOLD, fontSize); // 创建 Font 对象
15      FontMetrics fm = getFontMetrics(font); // 获取 FontMetrics 字体规格对象
16          fheigth = fm.getHeight();
17          baseLine = getSize().height / 2 + fheigth / 3;
18          twidth = fm.stringWidth(text);
19          w = fm.stringWidth(text);
20          w = (getSize().width - w) / 2;
21          position = w;
22          setBackground(Color.black);             // 设置小程序的背景色为黑色
23      }
24      public void start()
25      {
26          if (thread == null) {
27              thread = new Thread(this);          // 创建多线程
28              thread.start();                     // 启动多线程
29          }
30      }
31      public void stop()
32      {
33          thread.stop();                          // 停止运行线程
34          thread = null;
35      }
36      public void run()                           // 运行线程
37      {
38          while (true) {
39              repaint();                          // 重绘此组件
40              try {
41                  thread.sleep(30);
42              }                                   // 线程休眠
43              catch (InterruptedException e) {
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
44                }
45            }
46        }
47        public void update(Graphics g)                          // 更新组件
48        {
49            paint(g);
50        }
51        // 利用 clipRect()方法，每次调用显示方法 paint()时，
52        // 先用红色的笔画一遍文字，再用白色在裁剪区中画一遍文字
53        public void paint(Graphics g) {
54            g.setFont(font);
55            g.setColor(Color.red);
56            g.drawString(text, w, baseLine);                    // 第一遍显示
57            g.clipRect(position, 0, tsize, getSize().height);   // 设置裁剪区域
58            g.setColor(Color.white);
59            g.drawString(text, w, baseLine);                    // 第二遍显示
60            position = (position + 1) % (twidth + 100);         // 移动光标位置
61        }
62    }
```

【运行结果】程序运行结果如图 24-3 和图 24-4 所示。



图 24-3　灯光艺术字效果



图 24-4　灯光艺术字效果

【代码解析】首先在初始化方法 init()中设置显示的字体（第 10～23 行）。在 paint()方法中通过 Graphics 类的 getFont()或 setFont()可以获取或设置当前使用的字体（第 53～61 行）。run()方法为运行线程，设置线程的休眠时间为 30ms（第 36～46 行）。

2. 模拟字体展开效果

该示例模拟字体从最小慢慢展开，一直扩展到占据整个窗口，然后停止。

【范例 24-3】示例代码 24-3 是一个字体展开示例。

示例代码 24-3

```
01  public class WordsToExpand extends Applet implements Runnable
02      {
03      private Image image;                    //声明 Image 图像对象
04      private Graphics graphic;               //声明 Graphics 对象
05      private int width = 0, height = 0;      //声明高和宽变量
06      private String words;                   //定义文本内容
07      private Thread thread;                  //声明 Thread 多线程对象
08      private int xPosition = 0, yPosition = 0, myHeight;//定义 x 和 y 坐标点
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
09      private int times = 0;                          //表示线条出现的方向
10      private Font font;
11      public void init()                              //初始化 Applet 小程序
12      {
13          this.setSize(300, 200);
14          font = new Font("TimesRoman", Font.ROMAN_BASELINE, 30);
15          width = 300;
16          height = 200;
17          myHeight = height / 3;
18          yPosition = myHeight;
19          words = "这是 Applet 小程序";
20          image = createImage(width, height);
21          graphic = image.getGraphics();
22      }
23      public void start()                             //启动 Applet 小程序和多线程
24      {
25          if (thread == null)
26          {
27              thread = new Thread(this);
28              thread.start();
29          }
30      }
31      public void update(Graphics g)                  //更新组件
32      {
33          paint(g);
34      }
35      public void paint(Graphics g)                   //绘制组件
36      {
37          g.drawImage(image, 0, yPosition, width, myHeight, this);
38      }
39      public void run()                               //运行多线程
40      {
41          try {
42              while (true) {
43                  yPosition = 0;
44                  myHeight = height;
45                  graphic.setColor(Color.pink);
46                  graphic.fillRect(0, 0, width, height);
47                  repaint();
48                  thread.sleep(100);
49                  if (times == 0) {
50                      graphic.setColor(Color.red);
51                      for (int i = width; i >= 0; i--) {
52                      graphic.fillRect(i, height / 3, width, height / 10);
53                          repaint();
54                          thread.sleep(10);
55                      }
56                  } else if (times == 1) {
57                      graphic.setColor(Color.black);
58                      for (int i = 0; i <= width; i++) {
59                          graphic.fillRect(0, height / 3, i, height / 10);
60                          repaint();
61                          thread.sleep(10);
62                      }
63                  }
64                  yPosition = height / 3;
65                  myHeight = height / 3;
66                  for (int i = height / 3; i >= 0; i--) {
67                      xPosition = 0;
68                      yPosition--;
69                      myHeight = myHeight + 2;
70                      if (times == 0)//0 表示从右向左移动
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
71                          {
72                              graphic.setColor(Color.black);
73                              graphic.fillRect(0, 0, width, height);
74                              graphic.setFont(font);
75                              graphic.setColor(Color.yellow);
76                              graphic.drawString(words, 10, 35);
77                              times++;
78                          } else if (times == 1)          //1 表示从左向右移动
79                          {
80                              graphic.setColor(Color.red);
81                              graphic.fillRect(0, 0, width, height);
82                              graphic.setFont(font);
83                              graphic.setColor(Color.black);
84                              graphic.drawString(words, 10, 35);
85                              times = 0;
86                          }
87                          repaint();
88                          thread.sleep(100);
89                      }
90                      thread.sleep(2500);
91                      System.out.println(times);
92                  }
93              } catch (InterruptedException e) {
94              }
95          }
96      }
```

【代码解析】程序运行结果如图 24-5 和图 24-6 所示。

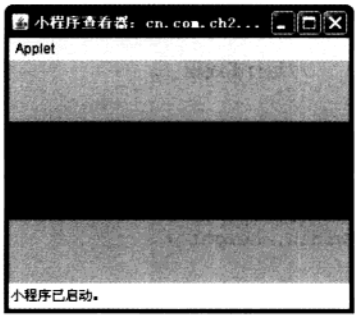


图 24-5　字体正在展开



图 24-6　已经完全展开的效果

【代码解析】以上程序中，首先在 init()方法中设置所要显示的字体（第 11～22 行）。在 paint()方法中通过 Graphics 类的 drawImage()方法输出 Applet，Graphics 类的 drawImage()方法用来显示 Image 对象（第 35～38 行）。run()方法为设置线程，根据时间来设置文字的移动及其大小（第 39～93 行）。

3．动态彩色文字

在 LED 显示屏上经常可以看到动态显示的文字，不过一般都是纯色的，Applet 可以模拟出动态的文字，文字的颜色也会不断地变化，而且文字的颜色是七彩的。

【范例 24-4】示例代码 24-4 是一个动态字体示例。

示例代码 24-4

```
01  public class DynamicFont extends Applet implements Runnable {
02      String str = null;
03      Thread thread = null;                    // 设置一个线程
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
04      char[] ch;
05      int p = 0;
06      Image image;
07      Graphics gphics;
08      Color[] color;
09      int one = 1;
10      int two = 18;
11      int three = 18;
12      private Font f;                              // 字体
13      private FontMetrics fm;                      // 字模
14      public void init() {
15          str = "这是Applet 程序";                  // 设置七彩文字内容
16          this.setSize(520, 200);                  // 设置 Applet 的大小
17          setBackground(Color.white);              // 设置背景颜色
18          ch = new char[str.length()];
19          ch = str.toCharArray();                  // 将字符串中的各个字符保存到数组中
20          image = createImage(getSize().width, getSize().height);
21          gphics = image.getGraphics();
22          f = new Font("", Font.BOLD, 18);
23          fm = getFontMetrics(f);                  // 获得指定字体的字体规格
24          gphics.setFont(f);                       // 设置组件的字体
25          float hue;
26          color = new Color[str.length()];         // 颜色的色元
27          for (int i = 0; i < str.length(); i++) {
28              hue = ((float) i) / ((float) str.length());
29          color[i] = new Color(Color.HSBtoRGB(hue, 0.8f, 1.0f)); // 颜色分配
30          }
31      }
32      public void start() {                        // 线程开始的类
33          if (thread == null) {                    // 如果线程为空,
34              thread = new Thread(this);           // 开始新的线程
35              thread.start();                      // 开始
36          }
37      }
38      // 终止线程
39      public void stop() {
40          if (thread != null) {                    // 如果线程不为空,
41              thread.stop();                       // 终止线程,
42              thread = null;                       // 使它为空
43          }
44      }
45      // 运行线程
46      public void run() {
47          while (thread != null) {
48              try {
49                  thread.sleep(200);               // 让线程沉睡 200ms
50              } catch (InterruptedException e) {
51              }
52              repaint();                           // 重新绘制界面
53          }
54      }
55      public void update(Graphics g) {    // 重写 update 方法，解决闪烁问题
56          int x, y;
57          double a;
58          gphics.setColor(Color.black);
59          gphics.fillRect(0, 0, getSize().width, getSize().height);
60          p += one;
61          p %= 7;                          // 主要控制字的速度，被除数越小，速度越快
62          // System.out.println(p+" p1");
63          for (int i = 0; i < str.length(); i++) {
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
64                // 主要控制弧度，被除数越小，弧度越大
65                a = ((p - i * one) % 7) / 4.0 * Math.PI;
66                // 求 x 坐标值
67                x = 30 + fm.getMaxAdvance() * i + (int) (Math.cos(a) * two);
68                y = 80 + (int) (Math.sin(a) * three);  // 求 y 坐标值
69                gphics.setColor(color[(p + i) % str.length()]);
70                gphics.drawChars(ch, i, 1, x, y);
71            }
72            paint(g);
73        }
74        public void paint(Graphics g) {
75            g.drawImage(image, 0, 0, this);
76        }
77    }
```

【运行结果】程序运行结果如图 24-7 所示。

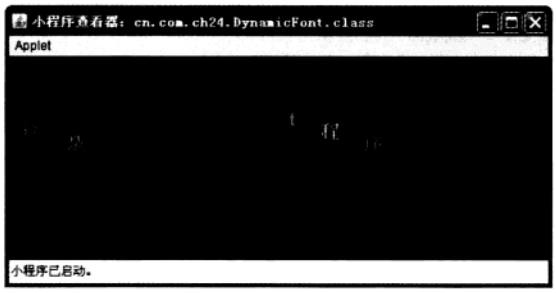


图 24-7　动态彩色文字

24.2.2　动态物体输出

在 Applet 小程序中，不仅能对静态的文字设置效果，还可以直接模拟动态的物体。

【范例 24-5】示例代码 24-5 是一个气泡示例。

示例代码 24-5

```
01  public class BubbleTest extends Applet implements Runnable
02  {
03      Thread artist = null;
04      int bubble = 0, thisbubble = 0;              // 气泡数量，当前气球编号
05      int MAXBUBBLES = 25;                         // 最大气泡数量
06      int stepper = 4;                             //移动的偏移量
07      int record[][] = new int[MAXBUBBLES][5];     // 记录气球的二维数组
08      public void init() {                         // 初始化 Applet
09          resize(400, 400);                        // 设定 Applet 尺寸
10      }
11      // 输出气泡
12      public void draw_bubble(int x, int y, int r, Color col, Graphics g)
13      {
14          int i;
15          for (i = x - r; i <= x + r; i++) {       // 输出一个圆形
16              g.setColor(col);
17              g.drawLine(i, y - (int) (Math.sqrt(r * r - ((i - x) * (i - x)))),
18                      i, y + (int) (Math.sqrt(r * r - ((i - x) * (i - x)))));
19          }
20      }
21      //设置气泡的移动
22  public void move_bubble(int x, int y, int r, Color col, int step, Graphics g)
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
23      {
24          int i;
25          for (i = x - r; i <= x + r; i++) { // 设置输出气球的上半部分
26              g.setColor(col);
27              g.drawLine(i, y - (int) (Math.sqrt(r * r - ((i - x) * (i - x)))),
28                      i, y + step
29                              - (int) (Math.sqrt(r * r - ((i - x) * (i - x)))));
30          }
31          for (i = x - r; i <= x + r; i++) { // 设置输出气球的下半部分
32              g.setColor(Color.white);
33              g.drawLine(i, y + (int) (Math.sqrt(r * r - ((i - x) * (i - x)))),
34                      i, y + step
35                              + (int) (Math.sqrt(r * r - ((i - x) * (i - x)))));
36          }
37      }
38      public void paint(Graphics g)
39      {
40          int i, j, tmp;
41          if (bubble < MAXBUBBLES || thisbubble < MAXBUBBLES) {
42              record[thisbubble][0] = (int) (Math.random() * 300);
43              record[thisbubble][1] = 320;
44              record[thisbubble][2] = (int) (Math.random() * 400) / 20;
45              record[thisbubble][3] = (int) (Math.random() * 255);
46              record[thisbubble][4] = (int) (Math.random() * 255);
47              // 输出气泡
48              draw_bubble(record[thisbubble][0], record[thisbubble][1],
49                  record[thisbubble][2], new java.awt.Color(
50                      record[thisbubble][3], record[thisbubble][4], 255),
51                      g);
52              // 如果气泡数小于最大值，则总气球数自增 1，当前气球编号自增 1
53              if (bubble < MAXBUBBLES) {
54                  bubble++;
55                  thisbubble++;
56              } else
57                  // 气泡数等于最大值
58                  thisbubble = MAXBUBBLES;
59          }
60          for (i = 0; i < bubble; i++) {
61              if (i % 5 <= stepper) {
62                  record[i][1] -= 1;
63                  // 移动气泡
64              move_bubble(record[i][0], record[i][1], record[i][2],
65                  new java.awt.Color(record[i][3], record[i][4], 255), 1,
66                      g);
67                  for (j = 0; j < i; j++) {
68                  tmp = ((record[i][1] - record[j][1])
69                      * (record[i][1] - record[j][1]) + (record[i][0] -
                           record[j][0])
70                              * (record[i][0] - record[j][0]));
71                  if (j != i && Math.sqrt(tmp) < record[i][2] + record[j][2])
72                      {
73                      for (tmp = record[i][2]; tmp >= -1; tmp = tmp - 2)
74                          draw_bubble(record[i][0], record[i][1],
75                                  record[i][2] - tmp, Color.white, g);
76                  draw_bubble(record[j][0], record[j][1], record[j][2],
77                          new java.awt.Color(record[j][3], record[j][4],
78                                  255), g);
79                  record[i][1] = -1;
80                  record[i][2] = 0;
81                  }
82              }
83          }
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
84                  if (record[i][1] + record[i][2] < 0 && bubble >= MAXBUBBLES) {
85                      thisbubble = i;
86                  }
87                  stepper = (int) (Math.random() * 10);
88              }
89          }
90          public void update(Graphics g)
91          {
92              paint(g);
93          }
94          public void start()
95          {
96              // 启动 applet,创建并启动线程
97              if (artist == null) {
98                  artist = new Thread(this);
99                  artist.start();
100             }
101         }
102         public void stop()
103         {
104             // 结束 applet
105             artist = null;
106         }
107         public void run()
108         {
109             // 启动线程
110             while (artist != null) {
111                 try {
112                     Thread.sleep(200);
113                 } catch (InterruptedException e) {
114                 }
115                 repaint();
116             }
117             artist = null;
118         }
119 }
```

【运行结果】程序运行结果如图 24-8 和图 24-9 所示。



图 24-8　开始冒泡



图 24-9　气泡增多

每当过年的时候，家家户户都会放鞭炮，放烟花。对于烟花的动态效果，使用 Applet 也可以模拟出来，但和前面气泡的模拟有些不同。

【范例 24-6】示例代码 24-6 是一个烟花示例。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



示例代码 24-6

```
    public class Fireworks extends Applet implements Runnable{
01  public int speed, variability, Max_Number, Max_Energy, Max_Patch,
02              Max_Length, G;
03      public String sound;
04      private int width, height;      // 获取当前容器边界的宽和高
05      private Thread thread = null;  // 设置线程
06      private BeaClassDemo bcd[];    // 创建 BeaClassDemo 类数组 bcd
07      public void init() {           // Applet 初始化
08          int i;
09          this.setSize(400, 400);    // 设置当前容器的宽和高
10          width = getSize().width - 1;
11          height = getSize().height - 1;
12          speed = 30;                                  // 烟花绽放的速度
13          variability = 10;
14          Max_Number = 100;                            // 可发出烟花的最大数目
15          Max_Energy = width + 50;
16          Max_Patch = 80;                              // 最大的斑点数
17          Max_Length = 200;                            // 斑点的最大距离
18          G = 50;                                      // 向地面弯曲的力度
19          bcd = new BeaClassDemo[Max_Number];     // 初始化 BeaClassDemo 数组
20          for (i = 0; i < Max_Number; i++)
21              // 创建 BeaClassDemo 类对象
22              bcd[i] = new BeaClassDemo(width, height, G);
23      }
24      public void start() {                           // 启动线程
25          if (thread == null) {
26              thread = new Thread(this);
27              thread.start();
28          }
29      }
30      public void stop() {                            // 停止线程
31          if (thread != null) {
32              thread.stop();
33              thread = null;
34          }
35      }
36      public void run() {
37          int i;
38      int E = (int) (Math.random() * Max_Energy * 3 / 4) + Max_Energy / 4 + 1;
39          int P = (int) (Math.random() * Max_Patch * 3 / 4) // 设置烟花的斑点数
                + Max_Patch / 4 + 1;
40  int L = (int) (Math.random() * Max_Length * 3 / 4)   // 设置烟花可发射出的距离
                + Max_Length / 4 + 1;
41          long S = (long) (Math.random() * 10000);    // 设置随机产生烟火的数量
42          boolean sleep;                               // 设置体眠的标志
43          Graphics g = getGraphics();
44          URL u = null;
45          while (true) {
46              try {
47                  thread.sleep(1000 / speed);
48              } catch (InterruptedException x) {
49              }
50              sleep = true;
51              for (i = 0; i < Max_Number; i++)
52                  sleep = sleep && bcd[i].sleep;
53              if (sleep && Math.random() * 100 < variability) {
54              E = (int) (Math.random() * Max_Energy * 3 / 4) + Max_Energy /
                4+ 1;
55  P = (int) (Math.random() * Max_Patch * 3 / 4) + Max_Patch / 4
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
                                + 1;
L = (int) (Math.random() * Max_Length * 3 / 4) + Max_Length / 4
                                + 1;
                        S = (long) (Math.random() * 10000);
                    }
                    for (i = 0; i < Max_Number; i++) {
                        if (bcd[i].sleep && Math.random() * Max_Number * L < 1) {
bcd[i].init(E, P, L, S);
bcd[i].start();
                        }
                        bcd[i].show(g);
                    }
                }
            }
            public void paint(Graphics g) {                    // 绘制组件
                g.setColor(Color.black);                       // 设置背景颜色为黑
                g.fillRect(0, 0, width + 1, height + 1);       // 根据参数画矩形
            }
}
class BeaClassDemo {
public boolean sleep = true;
private int energy, patch, length, width, height, G, Xx, Xy, Ex[], Ey[], x,
                    y, Red, Blue, Green, t;
            private Random random;                             // 声明 Random 类对象
            //类 BeaClassDemo 的构造方法
            public BeaClassDemo(int a, int b, int g)
            {
                width = a;
                height = b;
                G = g;
            }
            //初始化
            public void init(int e, int p, int l, long seed)
            {
                int i;
            // 赋值运算
                    energy = e;
                    patch = p;
                    length = l;
                // 创建一个带种子的随机数生成器
                random = new Random(seed);
                Ex = new int[patch];              // 初始化 int 数组 Ex，其长度为 patch
                Ey = new int[patch];              // 初始化 int 数组 Ey，其长度为 patch
                // 随机生成不透明的 RGB 颜色值
                Red = (int) (random.nextDouble() * 128) + 128;
                Blue = (int) (random.nextDouble() * 128) + 128;
                Green = (int) (random.nextDouble() * 128) + 128;
                Xx = (int) (Math.random() * width / 2) + width / 4;
                Xy = (int) (Math.random() * height / 2) + height / 4;
                for (i = 0; i < patch; i++) {
                    Ex[i] = (int) (Math.random() * energy) - energy / 2;
                    Ey[i] = (int) (Math.random() * energy * 7 / 8) - energy / 8;
                }
            }
            public void start() {
t = 0;
sleep = false;
            }
            public void show(Graphics g) {      // 输出烟花
                if (!sleep)                     // 如果休眠状态为 false
                    if (t < length) {
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
                int i, c;
                double s;
                Color color;
                c = (int) (random.nextDouble() * 64) - 32 + Red;
                if (c >= 0 && c < 256)
                    Red = c;
                c = (int) (random.nextDouble() * 64) - 32 + Blue;
                if (c >= 0 && c < 256)
                    Blue = c;
                c = (int) (random.nextDouble() * 64) - 32 + Green;
                if (c >= 0 && c < 256)
                    Green = c;
                color = new Color(Red, Blue, Green);
                for (i = 0; i < patch; i++) {
                    s = (double) t / 100;
                    x = (int) (Ex[i] * s);
                    y = (int) (Ey[i] * s - G * s * s);
                    g.setColor(color);
                    g.drawLine(Xx + x, Xy - y, Xx + x, Xy - y);
                    if (t >= length / 2) {
                        int j;
                        for (j = 0; j < 2; j++) {
                            s = (double) ((t - length / 2) * 2 + j) / 100;
                            x = (int) (Ex[i] * s);
                            y = (int) (Ey[i] * s - G * s * s);
                            g.setColor(Color.black);
                            g.drawLine(Xx + x, Xy - y, Xx + x, Xy - y);
                        }
                    }
                }
                t++;
            } else {
                sleep = true;
            }
        }
    }
```

【运行结果】程序运行结果如图 24-10 所示。



图 24-10　烟花效果

24.2.3　图片处理

Applet 中的图片显示主要有两个步骤，第一是装载图像文件；第二是显示图像。Applet 中

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

提供了用于装载图像的两个方法。

```
public Image getImage(URL url)
public Image getImage(URL url,String name)
```

【范例 24-7】示例代码 24-7 是一个相册示例。

示例代码 24-7

```
01  public class PhotoAlbum extends Applet {
02      private Choice myChoice;                    // 声明 Choice 对象
03      private String[] myString1, myString2;      // 声明 String 类型的数组
04      private int totalPics;                      // 声明一个 int 型变量
05      private Image offI;                         // 声明 Image 对象
06      private Image[] img;                        // 声明一组 Image 对象数组
07      private Graphics offG;                      // 声明一个 Graphics 对象
08      private MediaTracker imagetracker;          // 声明一个 MediaTracker 对象
09      public void init() {                        // 对 Applet 小程序进行初始化
10          this.setSize(600, 450);
12          this.setLayout(null);
13          myChoice = new Choice();
14          myChoice.setBounds(10, 10, 290, 20);
15          totalPics = 8;
16          myString1 = new String[totalPics];
17          myString2 = new String[totalPics];
18          img = new Image[totalPics];
19          for (int i = 0; i < totalPics; i++) {
20              myString1[i] = new String("");
21              myString2[i] = new String("");
22          }
23          String s = new String("");
24          imagetracker = new MediaTracker(this);      // 加载图片
25          for (int i = 0; i < totalPics; i++) {
26              s = "第 " + (i + 1) + " 张照片";
27              myString1[i] = s;
28              System.out.println(myString1[i]);
29              myChoice.addItem(s);                    // 向下拉列表中增加选项
30              s = (i + 1) + ".jpg";
31              myString2[i] = s;
32              img[i] = getImage(getDocumentBase(), s);
33              imagetracker.addImage(img[i], 0);
34          }
35          try {
36              imagetracker.waitForID(0);
37          } catch (InterruptedException e) {
38          }
39          add(myChoice);
40      offI = createImage(getSize().width, getSize().height - 40);// 创建图像
41          offG = offI.getGraphics();
42          offI = img[0];
43          offG.drawImage(offI, 0, 0, this);
44          repaint();
45      }
46      public void paint(Graphics g) {
47          g.drawImage(offI, 10, 40, this);
48      }
49      // 使用 action()方法来获得 Java Applet 小程序运行时所发生的事件
50      public boolean action(Event e, Object o) {
51          if (e.target == myChoice) {
52              // String s = new String("");
53              offG.setColor(this.getBackground());
54              offG.fillRect(0, 40, getSize().width, getSize().height - 40);
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
55                  offI = img[myChoice.getSelectedIndex()];
56                  offG.drawImage(offI, 0, 0, this);
57                  repaint();
58              }
59              return true;
60          }
61      }
```

【运行结果】程序运行结果如图 24-11 和图 24-12 所示



图 24-11　第一张照片



图 24-12　第二张照片

24.2.4　放大镜

对于老年人来说，上了年纪眼睛就会花，有时候看报纸和书籍就要有一副老花镜，或者是手拿一个放大镜，有一些研究古文物的专家也经常会手拿一个放大镜，在 Windows XP 操作系统中也带有放大镜。下面就用 Applet 来模拟一个放大镜的效果。

【范例 24-8】示例代码 24-8 是一个放大镜示例。

示例代码 24-8

```
01  public class MagnifierTest extends Applet implements MouseMotionListener {
02      Graphics g;                                  // 声明 Graphics 对象
03      Image image;                                 // 声明 Image 对象
04      Image back_Image;                            // 声明 Image 对象
05      String name;                                 // 声明 String 对象
06      MediaTracker tracker;                        // 声明媒体跟踪器 tracker
07      int Glass_X = 0, Glass_Y = 0;                // 放大镜初始位置
08      int Glass_W = 100, Glass_H = 100;            // 放大镜的宽度、高度
09      int width, height;                           // 声明背景图片的宽度和高度
10      public void init() {                         // 初始化 applet
11          // 加载图片
12          this.setSize(370, 500);
13          g = getGraphics();
14          name = "aa.jpg";
15          tracker = new MediaTracker(this);
16          back_Image = getImage(getDocumentBase(), name);
17          image = createImage(250, 100);      // 设置放大后的图像大小
18          Graphics offg = image.getGraphics();
19          offg.drawImage(back_Image, 0, 0, this);
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
20          addMouseMotionListener(this);           // 添加鼠标事件侦听
21      }
22      public void mouseDragged(MouseEvent e) {    // 鼠标拖拽事件处理
23      }
24      public void mouseMoved(MouseEvent e) {      // 处理鼠标移动事件
25          // 通过鼠标位置设置放大镜的位置
26          reprintGlass(Glass_X, Glass_Y, e.getX(), e.getY());
27          // 设置放大镜的当前位置
28          Glass_X = e.getX();
29          Glass_Y = e.getY();
30          // 若放大镜溢出 applet 则进行调整
31          if (Glass_X > (width - Glass_W / 2))
32              Glass_X = width - Glass_W / 2;
33          if (Glass_Y > (height - Glass_H / 2))
34              Glass_Y = height - Glass_H / 2;
35          printGlass();                           // 调用自定义方法输出放大镜
36      }
37      void printGlass() {
38          Graphics temp = g.create();             // 复制 g 的一个实例
39          // 为 temp 限制一个矩形区域
40          temp.clipRect(Glass_X, Glass_Y, Glass_W, Glass_H);
41          temp.drawImage(back_Image, -Glass_X, -Glass_Y, width * 2, height * 2,
42                  null);                          // 输出放大后的图像
43          g.setColor(Color.black);                // 设置放大镜边框的颜色
44          // 输出放大镜边框
45          g.drawRect(Glass_X, Glass_Y, Glass_W - 1, Glass_H - 1);
46      }
47      // 清除已经画过的矩形框和放大的图像
48      void reprintGlass(int X, int Y, int new_X, int new_Y) {
49          Graphics temp = g.create(); // 同上
50          if (new_X <= X && new_Y <= Y) {
51              temp.clipRect(new_X, new_Y + Glass_H, Glass_W + X - new_X, Y
52                      - new_Y);
53              temp.drawImage(image, 0, 0, null);
54              temp = g.create();
55              temp.clipRect(new_X + Glass_W, new_Y, X - new_X, Glass_H + Y
56                      - new_Y);
57              temp.drawImage(image, 0, 0, null);
58          } else if (new_X > X && new_Y <= Y) {
59              temp.clipRect(X, new_Y + Glass_H, Glass_W + new_X - X, Y - new_Y);
60              temp.drawImage(image, 0, 0, null);
61              temp = g.create();
62              temp.clipRect(X, new_Y, new_X - X, Glass_H + Y - new_Y);
63              temp.drawImage(image, 0, 0, null);
64          } else if (new_X > X && new_Y > Y) {
65              temp.clipRect(X, Y, Glass_W + new_X - X, new_Y - Y);
66              temp.drawImage(image, 0, 0, null);
67              temp = g.create();
68              temp.clipRect(X, Y, new_X - X, Glass_H + new_Y - Y);
69              temp.drawImage(image, 0, 0, null);
70          } else {
71              temp.clipRect(new_X, Y, Glass_W + X - new_X, new_Y - Y);
72              temp.drawImage(image, 0, 0, null);
73              temp = g.create();
74              temp.clipRect(new_X + Glass_W, Y, X - new_X, Glass_H + new_Y - Y);
75              temp.drawImage(image, 0, 0, null);
76          }
77      }
78  //    该方法是判断 infoflags 参数是否已完全加载了图像，是则返回 false；否则返回 true
79      public boolean imageUpdate(Image img, int infoflags, int x, int y, int w,
80              int h) {
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
81   if (infoflags == ALLBITS) {
     // ALLBITS 指示现在已完成了一幅以前绘制的静态图像，并且可以其最终形式再次绘制它
82               width = back_Image.getWidth(this);
83               height = back_Image.getHeight(this);
84               image = createImage(width + Glass_W / 2, height + Glass_H / 2);
85               Graphics offg = image.getGraphics();
86               offg.setColor(Color.white);
87               offg.fillRect(0, 0, width + Glass_W / 2, height + Glass_H / 2);
88               offg.drawImage(back_Image, 0, 0, this);
89               repaint();
90               return false;
91           } else
92               return true;
93       }
94       public void paint(Graphics g) {
95           g.drawImage(back_Image, 0, 0, this);    // 输出背景图片
96           printGlass();                           // 画放大镜
97       }
98   }
```

【运行结果】程序运行结果如图 24-13 所示。



图 24-13　放大镜效果

24.3　习题

【习题 24-1】编写一个简单的 Applet 程序，显示自己的姓名、班级等信息。

【分析】该题主要考查 Applet 输出字符串。

【核心代码】

```
public class DrawString extends Applet {
   public void paint(Graphics g) {
      g.setFont(new Font("TimeRoman", Font.BOLD, 20));
      g.setColor(Color.black);
      g.drawString("姓名: George Li, 班级: 2010-F", 10, 40);
   }
}
```

【习题 24-2】编写一个简单的 Applet 程序，在显示区域画一个椭圆，用黑色填充。

【分析】该题主要考查 Applet。

【核心代码】

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
public class DrawOval extends Applet {
    public void paint(Graphics g) {
        g.setColor(Color.black);
        g.fillOval(0,0,getSize().width, getSize().height);
    }
}
```

【习题 24-3】编写一个简单的 Applet 程序，显示一张图片。

【分析】该题主要考查 Applet。

【核心代码】

```
public class ShowImage extends Applet {
    private Image img;
    public void init() {
        System.out.println(getCodeBase());
        img = getImage(getCodeBase(), "rose.jpg"); // 将图片放到getCodeBase()目录
    }
    public void paint(Graphics g) {
        g.drawImage(img, 40, 40, 300, 300, this);
    }
}
```

【习题 24-4】编写一个简单的 Applet 程序，响应鼠标的按键动作。

【分析】该题主要考查 Applet。

【核心代码】

```
public class Mouse extends Applet {
    private String s = "";
    public void paint(Graphics g) {
        g.drawString(s, 20, 20);
    }
    public void init() {
        this.addMouseListener(new MouseAdapter(){
            public void mousePressed(MouseEvent e) {
                s = "mouse pressed";
                repaint();
            }
            public void mouseReleased(MouseEvent e) {
                s = "";
                repaint();
            }
        });
    }
}
```

【习题 24-5】编写一个简单的 Applet 程序，将用户在键盘上输入的字符显示在文本框中。

【分析】该题主要考查 Applet。

【核心代码】

```
public class Keyboard extends Applet {
    TextField txt = new TextField(8);
    public void init() {
        this.setBackground(Color.cyan);
        this.setForeground(Color.blue);
        this.setFont(new Font("Bookman Old Style", Font.BOLD, 20));
        this.add(txt);
        this.addKeyListener(new KeyAdapter() {
            public void keyPressed(KeyEvent e) {
                char c = e.getKeyChar();
                txt.setText("" + c);
            }
        });
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
    }
}
```

【习题 24-6】编写一个简单的 Applet 程序，播放一段声音。

【分析】该题主要考查 Applet。

【核心代码】

```
public class PlaySound extends Applet {
    AudioClip ac;
    public void init() {
        System.out.println(getCodeBase());
        ac = this.getAudioClip(getCodeBase(), "msg.wav");
                                                    // 将声音放到 getCodeBase()目录
    }

    public void start() {
        System.out.println(ac.toString());
        ac.loop();
    }

    public void paint(Graphics g) {
        g.drawString("Sound Demo", 5, 15);
    }
}
```

【习题 24-7】编写一个简单的 Applet 程序，绘制五个同心圆，圆与圆间隔 10 像素。

【分析】该题主要考查 Applet。

【核心代码】

```
public class Circle extends Applet {
    int s = 50;
    int w = 100;
    public void paint(Graphics g) {
        for (int i = 0; i < 5 * 20; i += 20) {
            g.drawOval(s+i/2, s+i/2, w-i, w-i);
        }
    }
}
```

【习题 24-8】编写一个简单的 Applet 程序，读取一个文本文件，并将其显示出来。

【分析】该题主要考查 Applet。

【核心代码】

```
public class ReadText extends Applet {
    JTextArea tf = new JTextArea();
    StringBuffer c = new StringBuffer();
    public void init() {
        this.setLayout(new GridLayout(1, 1));
        tf.setLineWrap(true);
        this.add(tf, BorderLayout.CENTER);

        try {
            BufferedReader br = new BufferedReader(new InputStreamReader(new
                                FileInputStream("E:\\t.txt"), "GB2312"));
            String s;
            while ((s = br.readLine()) != null) {
                System.out.println("--");
                c.append(s);
            }
        } catch (FileNotFoundException e) {
            System.out.println();
        } catch (IOException e) {
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
            System.out.println();
        }
    }

    public void paint(Graphics g) {
        System.out.println(c);
        tf.setText(c.toString());
    }
}
```

【习题 24-9】编写一个简单的滚动字幕的 Applet 程序。

【分析】该题主要考查 Applet。

【核心代码】

```
public class ScrollText extends Applet implements Runnable {
    Thread t;
    boolean stop = false;
    String s = "Welcome to Java Applet.";
    int p = 70;
    public void init() {
        this.setBackground(Color.cyan);
        this.setForeground(Color.yellow);
        this.setFont(new Font("TimeRoman", Font.BOLD, 20));
    }
    public void start() {
        t = new Thread(this);
        t.start();
    }
    public void run() {
        while (!stop) {
            p -= 1;
            if (p < -69) {
                p = 70;
            }
            repaint();
            try {
                Thread.sleep(300);
            } catch (InterruptedException e) {
                System.out.println();
            }
        }
    }

    public void stop() {
        stop = true;
        t = null;
    }
    public void paint(Graphics g) {
        g.setColor(Color.black);
        g.drawString(s, p, 50);
    }
}
```

【习题 24-10】编写一个简单的 Applet 程序，实现一段文字由小变大的动画。

【分析】该题主要考查 Applet。

【核心代码】

```
public class MorphingText extends Applet implements Runnable {
    private Image image;
    private Graphics graphic;
    private Font font;
    private int fontSize;
    private String str;
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
    private Thread thread;
    private Boolean stop = false;
    public void init() {
        this.setSize(400, 300);
        image = createImage(400, 300);
        graphic = image.getGraphics();
        str = "欢迎使用 Java Applet。";
        font = new Font("TimeRoman", Font.BOLD, 8);
    }

    public void start() {
        if (thread == null) {
            thread = new Thread(this);
            thread.start();
        }
    }

    public void stop() {
        stop = true;
    }

    public void update(Graphics g) {
        paint(g);
    }

    public void paint(Graphics g) {
        graphic.setColor(Color.black);
        graphic.fillRect(0, 0, getSize().width, getSize().height);
        font = new Font("TimeRoman", Font.BOLD, fontSize);
        graphic.setFont(font);
        graphic.setColor(Color.white);
        FontMetrics fm = graphic.getFontMetrics(font);
        int fontHeight = fm.getHeight();
        int w;
        int baseLine = getSize().height / 2 + fontHeight / 2;
        w = fm.stringWidth(str);
        w = (getSize().width - w) / 2;
        graphic.drawString(str, w, baseLine -= 20);
        g.drawImage(image, 0, 0, this);
        fontSize++;
    }
    public void run() {
        while (! stop) {
            repaint();
            if (fontSize > getSize().height) {
                fontSize = 0;
            }
            try {
                Thread.sleep(100);
            } catch (InterruptedException e) {
            }
        }
    }
}
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第 3 篇　网络编程篇

第 25 章　网 络 编 程

目前计算机应用的发展不断普及，特别是个人电脑在普通家庭中的普及，因此，对于计算机之间交互通信技术的实现越来越为人们所接受，因此，也促使了网络的发展。本章就主要介绍基于网络的 Java 编程。



25.1　网络基本概念

计算机网络就是将分布在不同地域的计算机和专门的外部设备使用通信线路相互连成一个规模大、功能强的网络系统，从而使众多的计算机可以比较方便地进行通信和共享资源信息等。计算机网络犹如一张巨大的蜘蛛网，每一个节点就相当于一台计算机或是一个外部设备。

使用网络可以将网络上的用户和本机一起共享网络资源，例如，可以将本机上的磁盘共享出来，将打印机进行共享等。

计算机网络根据规模的大小可以分为局域网（Local Area NetWork）和广域网（Wide Area NetWork）两种。局域网通常是指在一个比较小的地理范围内的各种计算机网络设备相互连在一起的通信网络，其中可以包含有一个或者是多个子网，这种网络内的计算机通常可以通过一个统一的接口与外界相连，或者与外界完全隔离。例如，一间屋子或者是一个公司内部的网络都是局域网。

广域网的地理范围比较大，通常就是一个国家或者是一个洲，例如，Internet 就是广域网。广域网的目的就是让分布比较远的各个局域网相连接。



25.2　TCP/IP

TCP/IP 协议的开发人员将 Internet 分为五个层次，这五个层次统称为互联网分层模型或者是互联网分层参考模型。本节将主要介绍 TCP 协议和 IP 协议。

25.2.1　TCP/IP 与网络协议

TCP/IP 协议是计算机网络中使用最广泛的体系结构之一，该协议是以 TCP 和 IP 为基础的所有相关协议的集合。TCP/IP 协议分为以下五层。

- 物理层：相对应的是网络中的基本硬件，也是 Internet 物理基础，就是可以看得见、摸得着的硬件设备，例如，个人计算机和服务器等。物理层为这些设备的特性指定了规范，这些规范使得各个设备之间都可以相互连接和使用。
- 网络接口层：该层定义了将数据组成正确帧的规程以及在网络中传输帧的规程，该层负责封装 IP 数据报，并使其可以通过网络进行传送。
- 互联网层：定义了互联网中传输数据信息的格式，以及一些从一个用户中发出，中间通过一个或者是多个路由器“辗转”到另一个用户之间的转发机制。互联网层是整个体系结构中的关键部分。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



- 传输层：为两个用户之间建立和管理有效可靠的连接，是负责总体的数据传输和数据控制。传输层提供端到端交互数据的机制。
- 应用层：为应用进程提供服务，是开放系统的最高层，作用就是在实现多个系统应用进程相互通信的同时完成一系列业务处理所需要的服务。

IP 协议规定了通信双方在通信中所共同遵守的约定，它属于互联网层控制协议，可以使两台计算机之间进行通信。

25.2.2　传输控制层（TCP）

计算机通过安装 IP 软件来保证计算机之间的数据发送和接收，不过 IP 协议还没有解决数据分组在传输过程中可能会出现的问题，为此，还需要安装 TCP 协议来解决可能出现的问题。因此，TCP 被称为是一种端对端的协议，该协议为两台计算机之间的连接起到了比较重要的作用。TCP 协议会为两台计算机之间建立一个连接、发送数据、接收数据和终止连接。

TCP 协议使用的重发技术和拥塞机制向应用程序提供了一个可靠的通信连接，可以使应用程序自动去适应网络上的各种变化。

【范例 25-1】示例代码 25-1 是一个 TCP 服务器端示例。

示例代码 25-1

```
01  public class TCPServer {
02      public static void main(String[] args) {    // java 程序主入口处
03          try {
04              Server server = new Server(8080);  // 传入端口号实例化对象
05          } catch (Exception e) {                // 捕获异常
06              System.out.println("测试服务器端监听出错: " + e.getMessage());
07          }
08      }
09  }
10  class Server {                                 // Socket 服务器端
11      private int port;                          // 端口
12      public Server(int port) {                  // 带参数的构造方法进行初始化
13          this.port = port;
14          start();                               // 调用启动服务器端的方法
15      }
16      public String infoUpperCase(String line) { // 处理信息
17          return line.toUpperCase();             // 将字符串大写
18      }
19      public void start() {                      // 启动服务器端
20          try {
21              // 根据端口创建服务器端 Socket 对象
22              ServerSocket serverSocket = new ServerSocket(port);
23              // 显示连接信息
24              System.out.println("服务器已启动，监听端口号为: " + port);
25              System.out.println("正在等待客户端连接.........");
26              // 挂起等待客户的请求
27              Socket socketAccept = serverSocket.accept();
28              BufferedReader in = new BufferedReader(new InputStreamReader(
29              socketAccept.getInputStream()));        // 获取读取客户端的数据流
30              // 获取写入客户端的数据输出流，参数 true 表示自动刷新输出缓冲区
31              PrintWriter out = new PrintWriter(socketAccept.getOutputStream(),
32                      true);
33              out.println("服务器端连接成功.........");  // 向客户发送连接信息
34              out.println("输入 quit 断开与服务器的连接");
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
35                boolean done = false;
36                while (!done) {
37                    String line = in.readLine();    // 读取客户端每行的内容
38                    if (line == null) {             // 没有写则不读取
39                        done = true;
40                    } else {
41                        // 显示客户端发送的内容
42                        System.out.println("来自客户端的信息：" + line);
43                        // 信息处理
44                        String message = infoUpperCase(line);
45                        // 向客户端发送信息
46                        out.println("来自服务器端的信息：" + message);
47                        if (line.trim().equals("quit"))     // 退出判断
48                            done = true;
49                    }
50                }
51                socketAccept.close();                       // 关闭通信资源
52            } catch (Exception e) {                         // 捕获异常
53                System.out.println("启动服务器端出现错误：" + e.getMessage());
54            }
55        }
56    }
```

【运行结果】程序运行结果如下。

```
服务器已启动，监听端口号为：8080
正在等待客户端连接.........
来自客户端的信息：你好
```

【代码解析】以上程序中，在服务器类的构造方法中传入端口和调用本类的 start()方法（第 12～15 行）。在 start()方法中创建服务器端的 Socket 对象，服务器端 Socket 对象的 accept()方法创建客户端的 Socket 对象，挂起等待客户端的请求（第 21～35 行）。使用 readLine()方法来读取客户端输入的内容，如果输入的是“quit”，则退出循环，释放有关通信资源（第 36～50 行）。

前面介绍了服务器端的设置，只有服务器端还不能够进行正常的连接，还需要创建一个客户端，并且在客户端开启之前，首先需要运行服务器端，才可以进行正常的连接。

【范例 25-2】示例代码 25-2 是一个 TCP 客户端示例。

示例代码 25-2

```
01  public class TCPClient {
02      public static void main(String[] args) throws IOException {
03          Client c = new Client("127.0.0.1", 8080);
04          c.connect();
05      }
06  }
07
08  class Client {
09      String ip;
10      int port;
11      Socket server = null;
12      Boolean isConnected;
13
14      public Client(String ip, int port) {
15          this.ip = ip;
16          this.port = port;
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
17      }
18
19      public void connect() throws IOException {
20          Socket server = new Socket(ip, port);
21          BufferedReader in = new BufferedReader(new 22 InputStreamReader
            server.getInputStream()));
23          PrintWriter out = new PrintWriter(server.getOutputStream());
24          BufferedReader wt = new BufferedReader(new 25 InputStream-Reader
            (System.in));
26
27          if (server.isConnected()) {
28              isConnected = true;
29              System.out.println("服务器信息: " + in.readLine());
30          }
31          while (isConnected) {
32              System.out.println("服务器信息: " + in.readLine());
33              System.out.print("请输入>");
34              String str = wt.readLine();
35              if (str.equals("quit")) {
36                  isConnected = false;
37              }
38              out.println(str);
39              out.flush();
40          }
41
42          out.close();
43          in.close();
44          wt.close();
45          server.close();
46      }
}
```

【运行结果】程序运行结果如下。

```
服务器信息: 服务器端连接成功.........
服务器信息: 输入 quit 断开与服务器的连接
请输入>你好
服务器信息:来自服务器端的信息: 你好
请输入>quit
```

25.2.3　IP 应用

IP 协议提供了能够适应各种网络硬件的灵活性，对底层网络硬件几乎没有任何要求。在一个网络中，只要从一个地点向另一个地点传送二进制数据，就能够使用 IP 协议加入 Internet。如果想要在 Internet 上进行信息的交流，就必须遵守 IP 协议，为此，每一台计算机都必须运行 IP 软件，以便时刻发送和接收信息。

如果要获取本机的计算机名称以及 IP 地址，通过 InetAddress 类可获取本地的主机名，在这期间有可能会出现主机没有找到的异常，所以要使用 try-catch 将其捕获。InetAddress 类中有一些常用方法，具体如下。

- public static InetAddress getByName(String host)：如果给定了主机名，则可以根据系统的配置名称来返回 IP 地址所组成的数组。参数 host 是机器名，可以是 IP 地址或者是 DSN 域名。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

- public static InetAddress getByAddress(byte[] addr)：如果给定了原始的 IP 地址，返回一个 InetAddress 对象。Ipv4 地址字节数组的长度必须为 4B，IPv6 字节数组的长度必须为 16B。
- public static InetAddress getLocalHost()：本地主机的 IP 地址。
- public static InetAddress[] getAllByName(String host)：如果给定了主机名，可根据系统的配置名称服务返回 IP 地址所组成的数组。
- public static InetAddress getByAddress(String host, Byte[] addr)：可以根据提供的主机名和 IP 地址创建一个 InetAddress 对象。主机名可以是机器的域名，也可以是其 IP 地址的文本表示形式。注意：也不在主机名上执行有效性检查。
- public byte[] getAddress()：返回此 InetAddress 对象的原始 IP 地址，结果按网络字节顺序，即地址的高位字节位于 getAddress()[0] 中。
- public String getHostAddress()：返回 IP 地址字符串表示形式。
- public String getHostName()：获取此 IP 地址的主机名。

【范例 25-3】示例代码 25-3 是一个获取 IP 地址的示例。

示例代码 25-3

```
01  public class GetIP
02  {
03      public static void main(String args[])
04      {
05          InetAddress baidu = null;     //创建一个 InetAddress 用于存放百度的地址
06          InetAddress myIP = null;      //创建一个 InetAddress 用于存放本机地址
07          InetAddress []yahoo=null;     //创建一个 InetAddress 用于存放雅虎地址数组
08          //取得地址
09          try
10          {
11              //使用 getByName 获得 baidu 的 IP 地址
12              baidu = InetAddress.getByName("www.baidu.com");
13              //获得本机地址
14              myIP = InetAddress.getLocalHost();
15              //使用 getAllByName 获得 yahoo 的地址数组
16              yahoo=InetAddress.getAllByName("www.yahoo.com.cn");
17          }
18          catch(UnknownHostException e)
19          {
20            e.printStackTrace();
21          }
22          System.out.println(baidu);              //打印出百度地址
23          System.out.println(myIP);               //打印出本机地址
24          for(int i=0;i<yahoo.length;i++)         //打印出雅虎的地址数组
25              System.out.println(yahoo[i]);
26      }
27  }
```

【运行结果】程序运行结果如下。

```
www.baidu.com/202.108.22.142
zhaozhen/192.168.1.102
www.yahoo.com.cn/202.165.102.205
```

【代码解析】首先创建三个 InetAddress 对象，分别存放百度、本机和雅虎的地址（第 05～07 行）。然后调用相应的方法获取 IP 地址（第 11～16 行）。最后分别打印出所获取的 IP 地址

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



（第 22～25 行）。



注意：获取地址时必须要使用 try-catch 块包围起来，否则会抛出异常。

IP 地址是由 32 位组成的，一个 IP 地址由 4 个不同的数字组成，而一个数字又对应 8 字节。在 IP 地址的分配中，可以分为 A、B、C、D、E 五种类型。

【范例 25-4】示例代码 25-4 是一个获取 IP 地址的类型示例。

示例代码 25-4

```
01  public class IPTypeTest
02  {
03      public static int getVersion(InetAddress ia)
04      {
05          byte[] b = ia.getAddress();          //定义一个字节型的数组，并初始化
06          if (b.length == 4) {                 //判断这个数组的长度是否等于 4
07              return 4;                        //返回数值 4
08          } else if (b.length == 16) {
09              return 6;
10          } else {
11              return -1;
12          }
13      }
14      public static char getClass(InetAddress ia)
15      {
16          byte[] b = ia.getAddress();          //定义一个字节型的数组，并初始化
17          if (b.length != 4) {                 //判断数组的长度是否不等于 4
18              //如果不等于 4，则抛出一个非法参数异常
19              throw new IllegalArgumentException("no ipv6 address");
20          }
21          int firstByte = b[0];                //定义一个整型的变量
22          //整型的变量与十六进制码比较
23          if ((firstByte & 0x80) == 0) {
24              return 'A';
25          } else if ((firstByte & 0xc0) == 0x80) {
26              return 'B';
27          } else if ((firstByte & 0xe0) == 0xc0) {
28              return 'C';
29          } else if ((firstByte & 0xF0) == 0xE0) {
30              return 'D';
31          } else if ((firstByte & 0xF8) == 0xF0) {
32              return 'E';
33          } else {
34              return 'F';
35          }
36      }
37      public static void main(String[] args) {
38          InetAddress ia = null;
39          try {
40              ia = InetAddress.getByName("www.sina.com"); //给定一个网络域名
41          } catch (UnknownHostException e) {
42              e.printStackTrace();
43          }
44          int len;
45          char king;
46          len = getVersion(ia);       //调用方法求出地址的长度
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
47            System.out.println("地址长度为：" + len);
48            king = getClass(ia);        //调用方法求出地址的类型
49            System.out.println("地址的类型为：" + king);
50        }
51    }
```

【运行结果】程序运行结果如下。

```
地址长度为：4
地址的类型为：C
```

【代码解析】首先使用 InetAddress 对象调用 getByName 方法来给定一个网络域名（第 40 行）。解析出 IP 地址，并对该地址进行分析，存放数组来获取长度，得到 IP 地址的类型（第 46～49 行）。

在 Windows 操作系统中可以查找主机，通过获取 IP 地址，然后使用 isHostName 构造方法判断是否由圆点和数字组成。

【范例 25-5】示例代码 25-5 是一个查找主机示例。

示例代码 25-5

```
01  public class HostTest
02  {
03      private static String lookup(String host)
04      {
05          InetAddress ia;
06          byte[] b = null;            // 初始化一个 byte 类型的数组
07          try {
08              ia = InetAddress.getByName(host);
09          } catch (UnknownHostException e) {
10              return "没有找到主机！";
11          }
12          if (isHostName(host)) {    //判断是否有主机
13              String s = "";         //定义一个字符串
14              for (int i = 0; i < b.length; i++) {    //循环遍历出主机的字符串
15                  int c = b[i] < 0 ? (b[i] + 256) : b[i];
16                  s += c;
17                  if (i != b.length - 1) {
18                      s += ",";
19                  }
20              }
21              return s;                           //返回一个字符串
22          } else {
23              return ia.getHostName();            //返回一个字符串
24          }
25      }
26      private static boolean isHostName(String host)
27      {
28          char[] ch = host.toCharArray();         //定义一个字符数组，并初始化
29          for (int i = 0; i < ch.length; i++) {
30              if (!Character.isDigit(ch[i])) {
31                  if (ch[i] != '.') {
32                      return true;
33                  }
34              }
35          }
36          return false;
37      }
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
38      public static void main(String[] args)
39      {
40          if (args.length > 0) {
41              for (int i = 0; i < args.length; i++) {      // 使用命令行
42                  System.out.println(lookup(args[i]));
43              }
44          } else {
45              BufferedReader br = new BufferedReader(new InputStreamReader(
46                      System.in));
47              System.out.println();
48              while (true) {
49                  try {
50                      String s = br.readLine();
51          if (s.equals("exit")) {//判断如果字符串与“exit”相同，就结束当前的语句块
52                              break;
53                      }
54                      System.out.println(lookup(s));
55                  } catch (IOException e) {
56                      System.err.println(e);
57                  }
58              }
59          }
60      }
61  }
```

【运行结果】程序运行结果如下。

```
aa
没有找到主机!
192.168.1.1
192.168.1.1
```

【代码解析】以上程序中，首先用 lookup 方法来判断用户输入的是否为一个合格的 IP 地址，并返回相应的信息（第 03～24 行），使用一个静态的构造方法 isHostName 实现了 IP 跟踪，最终找到主机所在位置（第 26～37 行）。



25.3　Socket

在 Java 中，Socket 用于对 IP 地址和端口进行描述，通常也称为“套接字”。我们可以通过这些信息来实现与其他主机之间进行通信。通过 Socket 可以使应用程序向网络发出请求或者是接收网络请求。

25.3.1　Socket 类

使用 Socket 类来建立连接，首先要在服务器端指定一个用来等待连接的端口创建一个 ServerSocklet 示例，然后需要在客户端通过规定一个主机和端口号来创建一个 Socket 示例，并连接到服务器上。Socket 类的构造函数如表 25-1 所示。

表 25-1　Socket 构造函数

构造函数	说　明
public Socket(String host, int port)	创建一个流套接字，并将其连接到指定主机上的指定端口号。host 表示主机名，而 port 则用来指定端口号

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

续表

构造函数	说　明
public Socket(InetAddress address, int port)	创建一个流套接字，并将其连接到指定 IP 地址的指定端口号
public Socket(String host, int port, InetAddress localAddr, int localPort)	创建一个套接字，并将其连接到指定远程主机上的指定远程端口。Socket 还会绑定到所提供的本地地址和端口。其中的参数 host 表示远程主机名，或者为 null，表示回送地址；参数 port 表示远程端口；参数 localAddr 表示要将套接字绑定到的本地地址；参数 localPort 表示要将套接字绑定到的本地端口
public Socket(InetAddress address, int port, InetAddress localAddr, int localPort)	创建一个套接字，并将其连接到指定远程端口上的指定远程地址。Socket 还会绑定到提供的本地地址和端口。address 表示远程地址，或者为 null，表示回送地址；port 表示远程端口；localAddr 表示要将套接字绑定到的本地地址；localPort 表示要将套接字绑定到的本地端口

Socket 提供了一些方法进行输入/输出流管理，如表 25-2 所示。

表 25-2　输入/输出流管理方法

构造函数	说　明
public InputStream getInputStream()	返回此套接字的输入流
public void shutdownInput()	该方法将此套接字的输入流置为“流的末尾”，发送到套接字的输入流端的任何数据都将被确认，然后被丢弃。如果在套接字上调用 shutdownInput() 后从套接字输入流读取内容，则流将返回 EOF（文件结束符）
public OutputStream getOutputStream()	返回此套接字的输出流
public void shutdownOutput()	关闭此套接字的输出流。对于 TCP 套接字，任何以前写入的数据都将被发送，然后与 TCP 的正常连接终止。如果在套接字上调用 shutdownOutput() 后写入套接字输出流，则该流将抛出 IOException



注意：上述方法都声明了 IOException 异常，需要在程序中处理。

关闭 Socket 对象使用的方法是：public void close()，表示关闭此套接字。套接字被关闭后，便不可在以后的网络连接中使用（即无法重新连接或重新绑定）。若要使用，需要创建新的套接字。

该类还定义了一些方法获得和设置 Socket 对象的属性，常用的方法如下：

- public InetAddress getInetAddress()：返回此套接字的地址。
- public InetAddress getLocalIAddress()：获得套接字绑定的本地地址。
- public int getPort()：获得此套接字连接到的远程端口。

25.3.2　ServerSocket 类

在服务器端使用的是 ServerSocket，主要用于接收客户端传送来的对象。该类的主要构造函数如下：

- ServerSocket()：创建非绑定服务器套接字。
- ServerSocket(int port) ：创建绑定到特定端口的服务器套接字。
- ServerSocket(int port, int backlog) ：利用指定的 backlog 创建服务器套接字，并将其绑定到指定的本地端口号。
- ServerSocket(int port, int backlog, InetAddress bindAddr) ：使用指定的端口、侦听

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



backlog 和要绑定到的本地 IP 地址创建服务器。

ServerSocket 类提供的主要方法有：

- public Socket accept()：该方法的作用是侦听并接收到连接此套接字的连接。此方法在进行连接之前一直阻塞。
- public void close()：该方法的作用是关闭服务器套接字。
- public void bind(SocketAddress addr)：该方法的作用是将 ServerSocket 绑定到指定的地址，IP 地址和端口号都被封装在 addr 中。
- public InetAddress getInetAddress()：该方法的作用是返回此服务器套接字的本地地址。
- public int getLocalPort()：该方法的作用是返回此套接字进行侦听的端口。
- public void setSoTimeout(int timeout)：该方法的作用是启用带有指定超时值的 SO_TIMEOUT，以 ms（毫秒）为单位。

可以使用下面的语句获得客户端的请求：

```
Socket linksocket=myserver.accept();
```

获得连接后进行通信，然后使用下面的语句关闭服务器监听：

```
myserver.close();
```

25.3.3　Socket 应用

本节主要介绍一个简单的聊天室程序，用于实现聊天室的基本功能。该程序分为客户端和服务器端，服务器端可以接收多个连接，客户端可以连接服务器。客户端发送消息给服务器端，服务器接收消息，转发给所有的客户。

【范例 25-6】示例代码 25-6 是一个聊天室示例。

示例代码 25-6

```
01  public class ServerTest {
02       public ServerTest(int port) throws IOException  //构造函数，参数为端口号
03      {
04            ServerSocket server=new ServerSocket(port); //创建 ServerSocket
05             //不断循环接收客户端的连接
06            while(true)
07            {
08                 Socket conn=server.accept();           //监听客户端的连接
09            //将服务器端的输入流封装到 DataInputStream 类中，该类支持与机器无关的编码方式
10                 DataInputStream in=new DataInputStream(conn.getInputStream());
11                 String who=in.readUTF();               //读取用户信息
12                 System.out.print("Client
13  "+"(IP:"+conn.getInetAddress()+")"+who+" enter!"+"\n");
14                 //每次监听到连接请求就创建一个线程
15            ServerHander cn=new ServerHander(who,conn); //对该连接创建一个线程
16                 cn.start();                            //启动该线程
17            }
18       }
19       public static void main(String args[])throws IOException{
20       new ServerTest(9001);                            //初始化服务器端
21       }
22  }
23  class ServerHander extends Thread                     //线程类
24  {
25       Socket socket;                                   //Socket 对象
26       DataInputStream in;                              //数据输入流
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
27      DataOutputStream out;                                   //数据输出流
28      String who;
29      //保存所有的客户端对象
30  protected static Vector<ServerHander> clientlist=new Vector<ServerHander>();
31  public ServerHander(String name,Socket socket)throws IOException //构造函数
32  {
33      this.who=name;
34      this.socket=socket;
35      in=new DataInputStream(new BufferedInputStream(socket.getInputStream()));
36      out=new DataOutputStream(new BufferedOutputStream(socket.getOutputStream
        ()));
37  }
38      public void run()
39    {
40          try{
41              clientlist.addElement(this);              //添加当前对象到 vector
42              sendallclient("欢迎  "+who+"  进入聊天室");//发送新客户进入消息给客户
43              //无限的循环
44              while(true)
45              {
46                  String msg=in.readUTF();              //读取输入信息
47                  sendallclient(who+"  说:  "+msg);     //将输入信息发送给所有的人
48              }
49          }
50          catch(IOException e)
51          {
52              System.out.println("Client exit or error.");
53          }
54              finally{
55                  clientlist.removeElement(this);       //清除存放客户端的向量
56                  sendallclient(who+"  退出 !");
57                  try
58                  {
59                      socket.close();                   //关闭 Socket 对象
60                  }
61                  catch(IOException ex)
62                  {
63                      System.out.println("Connection has been closed");
64                  }
65              }
66      }
67      protected static void sendallclient(String msg) //给客户端传递信息的方法
68      {
69          synchronized(clientlist){                     //对客户端 vector 加锁
70              //获得 vector 的枚举对象
71              Enumeration<ServerHander> allclients=clientlist.elements();
72          //遍历访问每个客户端对象
73          while(allclients.hasMoreElements())
74          {
75              ServerHander serh=(ServerHander)allclients.nextElement();
76              try{
77                  serh.out.writeUTF(msg);               //向客户端发送信息
78                  serh.out.flush();                     //刷新输出流，保证内容输出
79                  }
80                  catch(IOException exc)
81                  {
82                      serh.interrupt();                 //终端线程
83                  }
84              }
85          }
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
86      }
87  }
```

ClientTest.java

```
01  public class ClientTest
02  {
03      public ServiceFrame sf;                    //客户端窗体
04      private Socket csocket;                    //Socket 对象
05      private DataInputStream in;                //输入流
06      private DataOutputStream out;              //输出流
07      public static void main(String args[])
08      {
09          String who="";
10          //获得输入流
11          BufferedReader in=new BufferedReader(new InputStreamReader(System. in));
12          //输出提示信息
13          System.out.print("请输入你的名字");
14          try
15         {
16            who=in.readLine().trim();                //读取登录名
17          }
18          catch(IOException e)
19          {
20              e.printStackTrace();
21          }
22          new ClientTest(who,"127.0.0.1",9001);   //调用 Client 构造函数
23       }
24      public ClientTest(String who,String server,int port)
25      {
26          sf=new ServiceFrame("客户端聊天窗口");     //创建客户端窗体
27          sf.sendFD.addKeyListener(new ActListener(this,sf));//添加键盘监听事件
28          sf.addWindowListener(new ExitListener(this));
29          try{
30          csocket=new Socket(server,port);         //创建 Socket 对象
31          //输入流
32          in=new DataInputStream(new BufferedInputStream(csocket.getInput-
            Stream ()));
33      //输出流
34      out=new DataOutputStream(new BufferedOutputStream (csocket.getOutput-
        Stream()));
35          out.writeUTF(who);                       //向输出流输出 who 信息
36          out.flush();                             //刷新输出流
37          while(true)
38      {
39                sf.showTA.append("--"+in.readUTF()+"\n");     //在窗体输出信息
40          }
41          }
42          catch(Exception e)
43          {
44              System.out.println("Server error!");
45              this.close();
46              System.exit(0);
47              }
48      }
49      protected void send(String msg)              //输出信息
50      {
51          try
52          {
53              out.writeUTF(msg);                   //向输出流写入信息
54              out.flush();                         //刷新输出流
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
55              }
56          catch(IOException e)
57          {   }
58          }
59          protected void close()                          //关闭客户端，释放各种资源
60          {
61              try{
62                  sf.dispose();
63                  out.close();
64                  in.close();
65                  csocket.close();
66              }catch(IOException ex)
67              {}
68          }
69      }
70      class ServiceFrame extends Frame                    //客户端窗体类
71      {
72          protected TextArea showTA;                      //消息显示区
73          protected TextField sendFD;                     //消息输入区
74          public ServiceFrame(String winnm)
75          {
76              super(winnm);                               //调用父类的构造函数
77              setLayout(new BorderLayout());              //设置窗体布局
78              add("North",showTA=new TextArea());
79              showTA.setEditable(false);
80              add("South",sendFD=new TextField());
81              pack();
82              show();                                     //显示窗体
83              sendFD.requestFocus();
84          }
85      }
86      class ActListener extends KeyAdapter                //自己定义适配器类
87      {
88          ClientTest client;                              //客户类对象
89          ServiceFrame sframe;                            //窗体对象
90          public ActListener(ClientTest c,ServiceFrame sf)     //构造函数
91          {
92              client=c;
93              sframe=sf;
94          }
95          public void keyPressed(KeyEvent e)              //事件监听事件
96          {
97              if(e.getKeyCode()==KeyEvent.VK_ENTER){
98                  client.send(sframe.sendFD.getText());
99                  sframe.sendFD.setText("");
100             }
101         }
102     }
103     class ExitListener extends WindowAdapter            //定义自己的适配器类
104     {
105         ClientTest client;
106         public ExitListener(ClientTest c)               //构造函数
107         {
108             client=c;
109         }
110         public void windowClosing(WindowEvent e)        //窗体关闭事件
111         {
112             client.close();
113             System.exit(0);
114         }
115     }
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

【运行结果】程序运行结果如图 25-1 和图 25.2 所示。

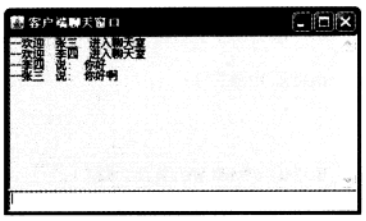


图 25-1　客户端 1 效果

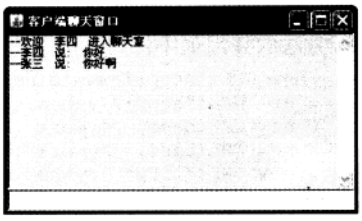


图 25-2　客户端 2 效果

25.4　URL

URL（Uniform Resource Locator）即统一资源定位符，表示在 Internet 上的某一个具体资源的访问地址。无论是寻址哪一种特定类型的资源或者是通过哪一种机制来获取某一些资源，URL 的基本语法是相同的，其格式如下。

```
<协议名>:<资源名>
```

对于协议名称来说，可以有多种选择，例如以下几种。

- file：指定的资源是本机上的文件。
- http：表示通过 HTTP 协议访问资源。
- mailto：通过 SMTP 访问指定电子邮件的资源。
- ftp：表示通过 FTP 协议访问资源。
- https：通过具有安全套接字层的 HTTP 协议访问资源。
- news：通过 NNTP 访问指定新闻地址的资源。

资源名的组成格式如下。

```
//<authority>:[<port>]/<path>[?<query>[#<fragment>]]
```

其中：

- authority：表示某台主机，也就是指定服务器的主机名或者是 IP 地址。
- port：表示端口号，是一个范围在 0～65 535 之间的整数。例如，HTTP 协议端口号为 80。
- path：表示路径，通常用于表示主机上的一个目录或文件。
- query：表示查询，用于给访问的目标资源传递参数。
- fragment：表示信息片段，是一个字符串，该部分用于指定目标资源中的片段。

25.4.1　URL 的使用

URL 常用的构造函数如下：

- public URL(String spec)：根据 String 创建 URL 对象。
- public URL(URL context, String spec)：通过在指定的上下文中对给定的 spec 创建 URL 对象。
- public URL(String protocol, String host, String file)：根据指定的 protocol（协议）名称、host（主机）名称和 file（文件）名称创建 URL。protocol 表示要使用的协议名称，host 表示主机名称，file 表示主机上的文件。该方法使用协议默认的端口，例如，端口 21 是 FTP 协议使用，23 是 Telnet 协议使用，25 是电子邮件使用，80 是 HTTP 协议使用等。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

- public URL(String protocol, String host, int port, String file)：根据指定 protocol（协议）、host（主机）、port（端口）号和 file（文件）创建 URL 对象。

以上构造方法的使用如下：

```
URL url1 =new URL("http://www.sina.com.cn/index.html");
URL base = new URL("http://yahoo.com.cn");
URL url2 = new URL(base, "mywork1.html");
URL url3 = new URL(base, "mywork2.html");
URL url5 = new URL("http", "yahoo.com.cn", 8080, "network.html");
```

在生成一个 URL 对象之后，它的属性是不能被改变的，但是可以通过几个比较常用的方法来获取这些属性值。常用方法如下：

- public String getProtocol()：获取该 URL 对象的协议名。
- public String getHost()：获取该 URL 对象的主机名。
- public String getPort()：获取该 URL 对象的端口号。
- public String getPath()：获取该 URL 对象的文件路径。
- public String getFile()：获取该 URL 对象的文件名。
- public String getRef()：获取该 URL 对象在文件中的相对位置。
- public String getQuery()：获取该 URL 对象的查询名。
- public final Object getContent()：获得此 URL 对象的内容。
- public String getAuthority()：获取此 URL 的授权。

【范例 25-7】示例代码 25-7 是一个聊天室示例。

示例代码 25-7

```
01  public class URLTest {
02      public static void main(String[] args) {
03          try
04          {
05             //创建一个 URL 对象
06               URL url=new URL("http://news.163.com/special/00014J0Q/nanjinge
                 xplode.html");
07          System.out.println("url 内容："+url.getContent());     //获取资源内容
08          System.out.println("主机："+url.getHost());            //获取资源的地址
09          System.out.println("端口："+url.getPort());       //获取资源的端口
10          System.out.println("查询名:"+url.getQuery());     //获取该资源的查询名
11          System.out.println("路径："+url.getPath());  //获取该资源的文件路径
12          System.out.println("文件名："+url.getFile());     //获取该资源的文件名
13          System.out.println("授权："+url.getAuthority()); //获取该资源的授权
14      }
15      catch (MalformedURLException e)
16      {
17          e.printStackTrace();
18      }
19      catch (IOException e)
20      {
21          e.printStackTrace();
22      }
23      }
24  }
```

【运行结果】程序运行结果如下。

```
url 内容：sun.net.www.protocol.http.HttpURLConnection$HttpInputStream@2e7263
主机：news.163.com
端口：-1
查询名:null
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
路径：/special/00014J0Q/nanjingexplode.html
文件名：/special/00014J0Q/nanjingexplode.html
授权：news.163.com
```

【代码解析】以上程序中，首先设定一个 URL 对象（第 06 行）。然后使用 URL 中的方法分别获取属性值并打印输出（第 07～13 行）。

25.4.2　提交表单信息

提交表单信息主要有两种方法，一种是以 GET 方式提交，另一种是以 POST 方式提交。前者是直接将数据的参数附在 URL 的尾部进行数据的传递；后者是从 URL-Connection 中获取输出流，然后将键值对写入输出流中，而不会将数据参数暴露在 URL 中。

GET 方式注册页面比较简单，但是这种方式存在许多缺点，例如，需要传递的数据会以明文方式直接显示在地址栏中，这样就对数据的安全性造成一定的影响。假设雅虎页面中有一个邮箱登录页面 E-mail，该页面接收 name 和 password 表单作为登录时的用户名和密码，这样，在地址栏中就会看到如下链接：

```
http://www.yahoo.com.cn/email?name=zhangsan&password=123456
```

使用 POST 的方式发送数据则不需要将数据放置在 URL 后面，所以也避免了一些不必要的安全问题。

25.4.3　简单的浏览器

本节将 Swing 和 URL 结合起来实现一个简单的浏览器。

```
01  public class HTMLTest extends JFrame
02  {
03      private JPanel contentPane;
04      private JPanel mainPanel=new JPanel();
05      //设置URL输入框
06      private JTextField urltext=new JTextField();
07      //设置页面显示
08      private JEditorPane htmlPanl=new JEditorPane();
09      public HTMLTest()
10      {
11          try{
12              jbInit();
13          }catch(Exception e){
14              e.printStackTrace();
15          }
16          //单击关闭按钮，自动退出程序并释放资源
17          this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
18      }
19      private void jbInit() throws Exception
20      {
21          contentPane=(JPanel)getContentPane();
22          //设置布局
23          contentPane.setLayout(new BorderLayout());
24          mainPanel.setLayout(new BorderLayout());
25          //设置提示
26          JLabel promptlabel =new JLabel("请输入网址");
27          urltext.setText("http://");
28          //设置监听器，输入完URL后按Enter键自动获取网页内容
29          urltext.addActionListener(new java.awt.event.ActionListener(){
30              public void actionPerformed(ActionEvent e) {
31                  setPaneContent(urltext.getText());
32              }
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
33            });
34            //设置 htmlPane 为不可编辑
35            htmlPanl.setEditable(false);
36            //添加事件监听器，当单击超链接时会自动获取链接内容
37        htmlPanl.addHyperlinkListener(new javax.swing.event.HyperlinkListener(){
38                public void hyperlinkUpdate(HyperlinkEvent e) {
39                    hyperlinkUpdate(e);
40                }
41            });
42            //添加滚动条
43            JScrollPane scrollpane=new JScrollPane();
44            scrollpane.getViewport().add(htmlPanl);
45            mainPanel.add(urltext,BorderLayout.NORTH);
46            mainPanel.add(scrollpane,BorderLayout.CENTER);
47            contentPane.add(promptlabel,BorderLayout.NORTH);
48            contentPane.add(mainPanel,BorderLayout.CENTER);
49            this.setSize(new Dimension(600,500));
50            this.setTitle("浏览器");
51            this.setVisible(true);
52        }
53        //将 URL 内容显示到 htmlPane 中
54        public void setPaneContent(String url)
55        {
56            try{
57                htmlPanl.setPage(url);
58            }catch(Exception e){
59                JOptionPane.showMessageDialog(this,"URL地址不正确:"+urltext.get
                  Text(),"输入不正确",0);
60            }
61        }
62        //将超链接内容显示到 htmlPane 中
63        public void hyperLinkUpdate(HyperlinkEvent e)
64        {
65    if(e.getEventType()== javax.swing.event.HyperlinkEvent.EventType.ACTIVATED){
66            URL url=e.getURL();
67            setPaneContent(url.toString());
68            }
69        }
70        public static void main(String[] args) {
71            new HTMLTest();
72        }
73    }
```

【运行结果】程序运行结果如图 25-3 所示。

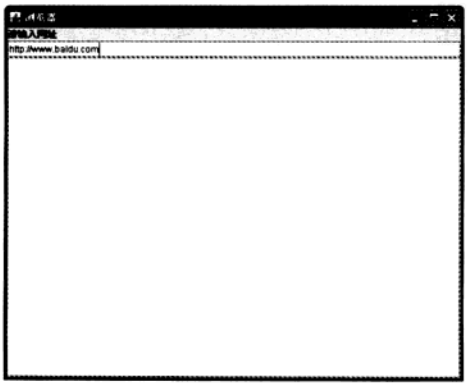


图 25-3　浏览器效果

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



25.5　习题

【习题 25-1】获取已经建立连接的远端给定主机和端口以及本地地址和端口的信息。

【分析】该题主要考查 TCP/IP 协议以及 IP 的应用。

【核心代码】

```
for (int i = 0; i < args.length; i++) {
    try {
        Socket s = new Socket(args[i], 80);                    //创建一个 Socket 对象
        //显示连接上主机的端口、本地地址、IP 等信息
        System.out.println("Connected to" + s.getInetAddress()
                + "on port" + s.getPort() + "from port"
                + s.getLocalPort() + "of" + s.getLocalAddress());
    } catch (UnknownHostException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
    }
```

【习题 25-2】编写一个程序，获取网络上某一台主机的 IP 地址。

【分析】该题主要考查 IP 的应用。

【核心代码】

```
try {
            InetAddress ind = InetAddress.getLocalHost();    //获取本机的 IP 地址
            System.out.println(ind);                         //将 IP 地址显示出来
        } catch (Exception e) {
            System.out.println("没有找到主机的地址！");
        }
```

【习题 25-3】编写一个服务器和客户端程序，实现服务器端和客户端之间的通话。

【分析】该题主要考查 Socket 类和 ServerSocket 类的应用。

服务器端：

```
public void start() {                                          // 启动服务器端
        try {
            // 根据端口创建服务器端 Socket 对象
            ServerSocket serverSocket = new ServerSocket(port);
            // 显示连接信息
            System.out.println("服务器已启动，监听端口号为：" + port);
            System.out.println("正在等待客户端连接.........");
            // 挂起等待客户的请求
            Socket socketAccept = serverSocket.accept();
            BufferedReader in = new BufferedReader(new InputStreamReader(
                    socketAccept.getInputStream()));    // 获取读取客户端的数据流
            // 获取写入客户端的数据输出流,true:自动刷新
            PrintWriter out = new PrintWriter(socketAccept.getOutputStream(),
                    true);
            out.println("服务器端连接成功.........");       // 向客户发送连接信息
            out.println("输入 quit 断开与服务器的连接");
            boolean done = false;
            while (!done) {
                String line = in.readLine();               // 读取客户端每行的内容
                if (line == null) {                        // 没有写则不读取
                    done = true;
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
                } else {
                    // 显示客户端发送的内容
                    System.out.println("来自客户端的信息: " + line);
                    // 信息处理
                    String message = infoUpperCase(line);
                    // 向客户端发送信息
                    out.println("来自服务器端的信息: " + message);
                    if (line.trim().equals("quit"))     // 退出判断
                        done = true;
                }
            }
            socketAccept.close();                              // 关闭通信资源
        } catch (Exception e) {                                // 捕获异常
            System.out.println("启动服务器端出现错误: " + e.getMessage());
        }
```

客户端:

```
public void connectServer() {                                  // 连接方法
        try {
            Socket socketConn;                                 // 声明 Socket 连接
            // 判断是否是本机 IP
            if (host.equals("localhost") || host.equals("127.0.0.1")) {
                // 创建本地 Socket 连接
                socketConn = new Socket(InetAddress.getLocalHost(), port);
            } else {                                           // 创建远程 Socket 连接
                socketConn = new Socket(InetAddress.getByName(host), port);
            }
            BufferedReader stdin = new BufferedReader(new InputStreamReader(
                    System.in));                               // 获得从键盘输入的流
            PrintWriter out = new PrintWriter(socketConn.getOutputStream(),
                    true);                                     // 获得服务器写内容的数据流
            BufferedReader in = new BufferedReader(new InputStreamReader(
                    socketConn.getInputStream()));// 获得接收服务器发送内容的缓冲流
            // 从服务器获得信息
            System.out.println("服务器信息: " + in.readLine());
            System.out.println("服务器信息: " + in.readLine());
            System.out.print("请输入>");                       // 用户输入
            boolean done = false;
            while (!done) {
                String line = stdin.readLine();                // 获得从键盘输入的每行字符
                out.println(line);                             // 发送到服务端
                if (line.equalsIgnoreCase("quit"))             // 读到 quit 则结束循环
                    done = true;
                String info = in.readLine();                   // 从服务器读取字符串
                System.out.println("服务器信息:" + info);// 显示从服务器发送来的数据
                if (!done)                                     // 用户输入
                    System.out.print("请输入>");
            }
            socketConn.close();                                // 关闭资源
        } catch (SecurityException e) {                 // 捕获安全性错误时引发的异常
            System.out.println("连接服务器出现安全问题! ");
        } catch (IOException e) {                       // 捕获 I/O 流异常
            System.out.println("连接服务器出 4、现 I/O 错误! ");
        }
```

【习题 25-4】编写一个程序用于扫描给定主机的 TCP 端口。运行程序，并在后面加主机地址作为参数，则可以获得此主机的各个 TCP 端口信息。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



【分析】该题主要考查的是传输控制层 TCP 的应用。

【核心代码】

```
for(int i=100;i<500;i++){                                    //使用循环扫描端口
            try {
                Socket s = new Socket(str,i);                //创建 Socket 对象
                System.out.println("There is a server on"+i+"of"+s);
            } catch (UnknownHostException e) {               //捕获找不到主机的异常
                e.printStackTrace();
                break;
            } catch (IOException e) {                        //捕获 I/O 流发生的异常
                e.printStackTrace();
            }
        }
```

【习题 25-5】编写一个程序获取输入主机名称的 IP 地址，并判断其 IP 地址类型。

【分析】该题主要考查 TCP/IP 协议以及 IP 的应用。

【核心代码】

```
public class IPType {
   public static void main(String[] args) {
      Scanner reader = new Scanner(System.in);
      System.out.println("请输入主机名称：");
      String n = reader.next();
      byte[] b = {};
      try {
         InetAddress ia = InetAddress.getByName(n);
         b = ia.getAddress();
      } catch (UnknownHostException e) {
         e.printStackTrace();
      }
      System.out.println("此主机地址的长度：" + b.length);
      String t = "";
      if (b.length == 4) {
         if ((b[0] & 0x80) == 0) {
            t = "A";
         } else if ((b[0] & 0xc0) == 0x80) {
            t = "B";
         } else if ((b[0] & 0xe0) == 0xc0) {
            t = "C";
         } else if ((b[0] & 0xF0) == 0xE0) {
            t = "D";
         } else if ((b[0] & 0xF8) == 0xF0) {
            t = "E";
         } else {
            t = "F";
         }
         System.out.println("此主机地址的类型：" + t);
      }
   }
}
```

【习题 25-6】编写一个程序，使用 Java 的 URL 类输出某一连接地址的各种属性。

【分析】该题主要考查 TCP/IP 协议以及 IP 的应用。

【核心代码】

```
public class URLProperty {
   public static void main(String[] args) {
      Scanner reader = new Scanner(System.in);
      System.out.println("请输入 URL 地址：");
      String n = reader.next();
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
            System.out.println(n + "的相关属性：");
            try {
               URL u = new URL(n);
               System.out.println("url："+ u.getContent());
               System.out.println("主机："+ u.getHost());
               System.out.println("端口："+ u.getPort());
               System.out.println("查询："+ u.getQuery());
               System.out.println("路径："+ u.getPath());
               System.out.println("文件："+ u.getFile());
               System.out.println("授权："+ u.getAuthority());
            } catch (MalformedURLException e) {
               e.printStackTrace();
            } catch (IOException e) {
               e.printStackTrace();
            }
         }
      }
```

【习题 25-7】打印出当前类的绝对路径和 CLASSPATH。

【分析】该题主要考查 URL 的应用。

【核心代码】

```
public class FileURL {
   public static void main(String[] args) {
      System.out.println(new File("E:\\t.txt").toURI());
      System.out.println(FileURL.class.getResource(""));
      System.out.println(FileURL.class.getResource("/"));
      System.out.println(FileURL.class.getClassLoader().getResource(""));
      System.out.println(Thread.currentThread().getContextClassLoader().
getResource(""));
      System.out.println(ClassLoader.getSystemResource(""));
   }
}
```

【习题 25-8】编写一个程序，根据给定的 URL 地址，判断此地址是否存在。

【分析】该题主要考查 URL 的应用。

【核心代码】

```
public class ValidURL {
   public static void main(String[] args) throws MalformedURLException {
      Scanner reader = new Scanner(System.in);
      System.out.println("请输入 URL：");

      URL url = new URL(reader.next());
      byte[] b = new byte[5000];
      try {
         int n = url.openStream().read(b, 0, 5000);
         if (n == -1) {
            System.out.println("此地址不存在");
         } else {
            System.out.println("这是一个合法的地址");
         }
      } catch (IOException e) {
         System.out.println("此地址不存在");
      }
   }
}
```

【习题 25-9】使用 URL 编写一个读取网页内容的爬虫程序。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



【分析】该题主要考查 TCP/IP 协议以及 IP 的应用。

【核心代码】

```
public class ReadHTML {
    public static void main(String[] args) throws IOException {
        URL url = new URL("http://www.baidu.com");
        URLConnection uc = url.openConnection();
        BufferedReader r = new BufferedReader(new InputStreamReader(uc. GetInput-
Stream(), "GB2312"));
        String s;
        while ((s = r.readLine()) != null) {
            System.out.println(s);
        }
        r.close();
    }
}
```

【习题 25-10】编写一个程序，使用 Socket 向服务器传送文件。

【分析】该题主要考查 Socket 的应用。

【核心代码】

```
public class SendFile {
    public static void main(String[] args) throws IOException {
        new Thread(new Server()).start();
        Client c = new Client();
        c.connect(1234);
        c.send(new File("E:\\t.txt"));
    }


    static class Client {
        private Socket s;
        public void connect(int port) {
            try {
                s = new Socket(InetAddress.getLocalHost(), port);
            } catch (IOException e) {
                e.printStackTrace();
            }
        }

        public void send(File f) throws IOException {
            byte[] b = new byte[1024];
            int n;
            DataOutputStream out = new DataOutputStream(new BufferedOutputStream(s.
getOutputStream()));
            FileInputStream in = new FileInputStream(f);
            while ((n = in.read(b)) != -1) {
                out.write(b, 0, n);
                out.flush();
            }
            in.close();
            out.close();
        }
    }

    static class Server implements Runnable {
        private ServerSocket server;
        private int port = 1234;
        private boolean stop = false;

        public void receive(Socket socket) throws IOException {
            byte[] b = new byte[1024];
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
                int n;
                DataInputStream in = new DataInputStream(new BufferedInputStream(socket.
getInputStream()));
                RandomAccessFile out = new RandomAccessFile(new File("E:\\_copy.txt"),
"rw");
                while ((n = in.read(b)) != -1) {
                    out.write(b, 0, n);
                    out.skipBytes(n);
                }
                stop = true;
                in.close();
                out.close();
            }

            public void run() {
                try {
                    server = new ServerSocket(port);
                } catch (IOException e) {
                    e.printStackTrace();
                }
                while (! stop) {
                    try {
                        Socket socket = server.accept();
                        if (null == socket) continue;
                        receive(socket);
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                }
            }
        }
    }
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第 26 章　JSP

JSP 是一种动态网页标准，类似 ASP 技术。JSP 技术在传统 HTML 文件的基础上添加 Java 程序段或 JSP 标记，从而形成动态的页面。

26.1　JSP 运行环境搭建

JSP 是运行在服务器端的，如果要运行它，就需要安装 JSP 引擎。比较常见的 JSP 引擎有 Tomcat、JRun 和 Resin 等。不过对于初学者来说，使用 Tomcat 就是一个比较不错的选择。本节就主要介绍 Tomcat 的下载和安装配置。

26.1.1　Tomcat 下载与安装

打开 Tomcat 的主页（http://tomcat.apache.org），选择页面左边的超链接“DownLoad”→“Tomcat 5.5”，如图 26-1 所示，进入下载页面后，选择“Windows Service Installer”版，选择这个版本可以在安装后直接将 Tomcat 注册为一个 Windows 服务。

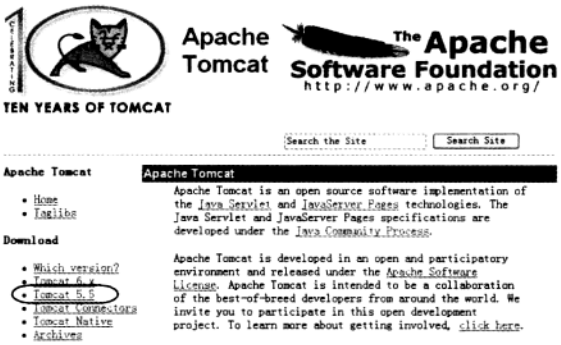


图 26-1　选择下载 Tomcat 版本

双击下载后的文件就可以开始安装，安装过程并不复杂，一般单击“Next”按钮即可，不过也有几个地方需要特别设定，具体步骤如下。

① 如图 26-2 所示，选择安装模块的向导页需要选中“Examples”复选框，完成后单击“Next”按钮。

② 在之后弹出的 Tomcat 安装向导页中，需要设置 Tomcat 的用户名、密码、端口号（如图 26-3 所示），密码则是设置 Tomcat 的超级管理员密码，用户名和密码都会记录在 Tomcat 安装目录下的“conf/tomcate-users.xml”文件中，并且可以在此文件中直接进行修改。在此页中，还需要设置 HTTP 监听端口，默认端口是 8080（如果这个端口已经被其他软件占用，可以另外选择一个空闲端口，如 8088）。

安装完成后，Tomcat 随即启动，可以进入 Windows 的服务窗口查看（如图 26-4 所示），Tomcat 的状态显示为“已启动”。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

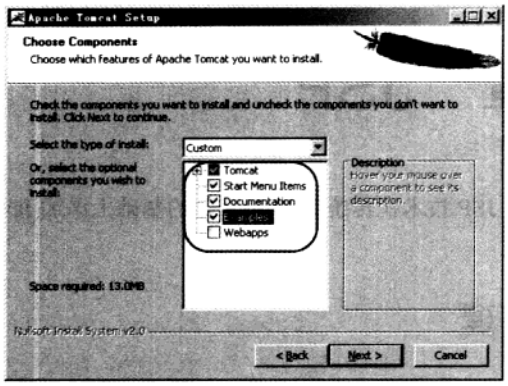


图 26-2　Tomcat 安装模块

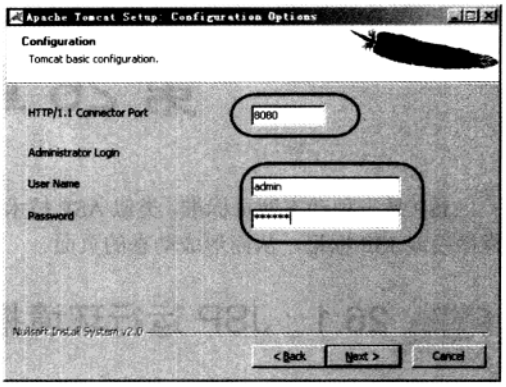


图 26-3　设置用户名、密码、端口号

在 IE 浏览器的地址栏中输入“http://127.0.0.1:8080”，其中 IP 地址 127.0.0.1 为本地地址，8080 为 Tomcat 安装时设置的端口号。Tomcat 安装成功后，会显示如图 26-5 所示的界面。

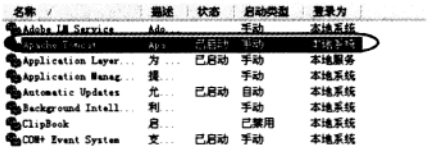


图 26-4　Tomcat 已启动

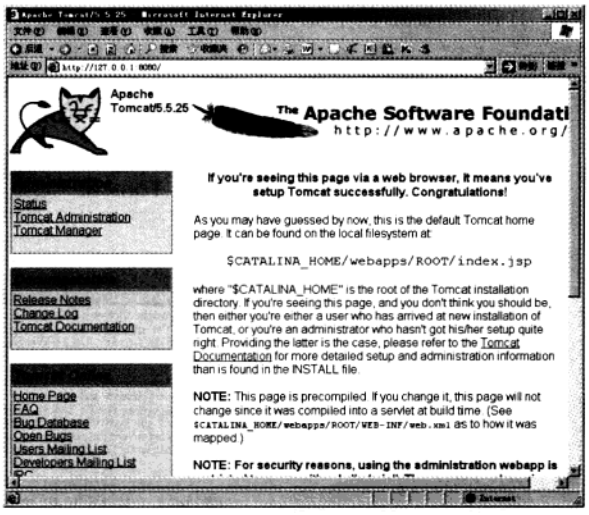


图 26-5　Tomcat 欢迎界面

在页面的左侧分别有 Tomcat Administration 和 Tomcat Manager 两个超链接，前者用于设置数据库连接池和管理用户权限，后者用于设置发布或取消 Web 应用程序。

单击 Tomcat Manager，会弹出用户名和密码验证（如图 26-6 所示），然后就可以进入 Tomcat 的管理界面对 Tomcat 进行管理。

26.1.2　Tomcat 的环境变量配置

安装完成 Tomcat 后，需要对其进行环境变量的设置，具体步骤如下：

① 鼠标右键单击“我的电脑”图标，选择“属性”选项，弹出“系统属性”对话框，单击“高级”标签页，在出现的对话框中单击“环境变量”按钮，弹出系统变量对话框，单击“新建”按钮，弹出新建系统变量对话框，在“变量名”对应的文本域中输入“TOMCAT_HOME”，

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

在“变量值”对应的文本域中输入 Tomcat 的安装目录，如图 26-7 所示。

② 单击“确定”按钮完成设置后，返回环境变量对话框中，选择已经设置的环境变量“classpath”，在变量值最后输入“%TOMCATE_HOME%\lib”，在“变量值”对应的文本域最后输入“;”。单击“确定”按钮完成设置，如图 26-8 所示。

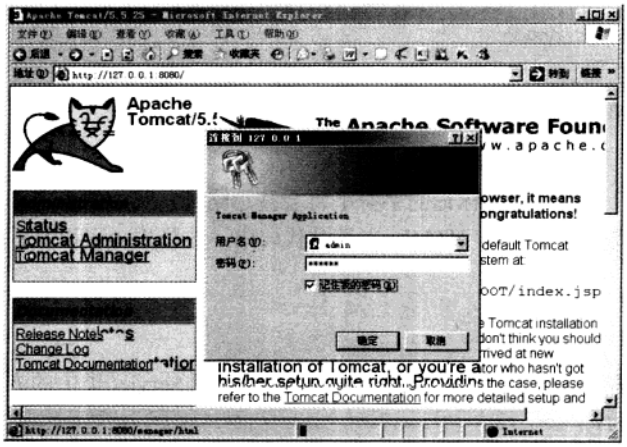


图 26-6　密码验证

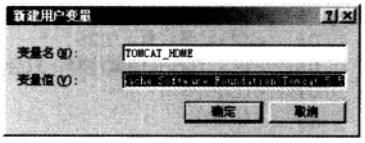


图 26-7　TOMCAT 设置环境变量

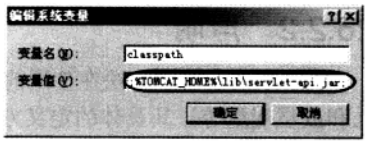


图 26-8　设置 classpath

26.2　基本语法

JSP 的页面就是由特定的 JSP 元素内嵌到 HTML 网页中形成的。除了 HTML 标签之外，JSP 中提供了五种构建网页内容时所需要的元素，具体包括指令、声明、程序代码、表达式和注释。

26.2.1　注释

HTML 注释指在客户端的源代码中显示的注释。HTML 的语法如下：

```
<!-- comment [ <%= expression %> ] -->
```

下面是 HTML 注释的例子。

```
<!--这个注释会显示在 HTML 源代码中 -->
```

在客户端的 HTML 源代码中能够看到同样的代码：

```
<!--这个注释会显示在 HTML 源代码中 -->
```

下面是另外一个 HTML 注释的例子。

```
<!--页面加载时间是：<%= (new java.util.Date()).toLocaleString() %> -->
```

客户端页面的 HTML 源代码中会看到：

```
<!--页面加载时间是：September 1,1708 -->
```

这些注释与在 HTML 源代码中写的注释相似。用户打开一个网页后，单击鼠标右键，在弹出的快捷菜单中选择“查看源代码”才能看到，但不同的页面可能产生不同的注释。在注释的表达式中，可以使用任何合法的表达式。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

隐藏注释只在 JSP 代码中显示，而不会将该注释发送给客户端。隐藏注释的语法如下：

```
<%-- 隐藏内容 --%>
```

下面是一个隐藏注释的例子。

```
01  <%@ page language="java" %>
02  <html>
03  <head><title>隐藏注释</title></head>
04  <body>
05  <h2>隐藏注释</h2>
06  <%-- 这些是隐藏注释，只会显示在代码中，不会发送到客户端 --%>
07  </body>
08  </html>
```

上面的 JSP 文件中，使用“<%--”和“--%>”标记的字符串将会被忽略掉，其中的内容不会被编译。所以不会显示在客户端浏览器和客户端 HTML 文件源代码中。注释内容可以是除“--%>”之外的任意内容。

26.2.2 声明

JSP 中的变量和方法在使用前都需要进行声明，JSP 声明用来定义插入到 Servlet 类中的方法和成员变量等，其具体的定义方法为：

```
<%!Java 声明%>
```

声明一般会结合 JSP 表达式或者 JSP 脚本一起使用，因为声明不会有任何输出。下面用一个示例来进行说明，在 MyEclipse 中 Web 项目的 WebRoot 目录下创建文件“declaration.jsp”，然后在其中输入以下代码。

【范例 26-1】示例代码 26-1 是一个声明示例。

示例代码 26-1

```
01  <%@ page language="java" import="java.util.*" pageEncoding="GBK"%>
02  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
03  <html>
04   <%!
05   private String name="张三";
06   private String Returnname(){
07   return name;
08   }
09    %>
10    <body>
11  <%
12  String name=Returnname();
13  out.println("姓名为："+name);
14   %>
15    </body>
16  </html>
```

【运行结果】运行效果如图 26-9 所示。

【代码解析】以上程序中，首先定义了一个字符串类型的 name（第 05 行）；然后定义一个方法，并返回 name 值（第 06、07 行）；最后调用该方法并打印 name 的值（第 12、13 行）。

如果在使用变量和方法之前没有声明，就会在运行时报错，JSP 中一次可以声明多个变量，但是它们之间需要使用“,”来分隔，并且以“;”结尾。声明时有以下几点需要注意。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



图 26-9　声明示例效果

- 声明必须以“;”结尾，这与 Scriptlet 规则相同。
- 可以直接使用由<% @ page %>包含的变量，不用重新声明。
- 一个声明的有效范围是这个页面内部。如果多个页面都需要同样的声明，可以将这些声明写成一个文件，然后使用<%@ include %>或<jsp:include >方法将文件包含进来。

26.2.3　表达式

表达式用于在 JSP 中输出变量或者输出结果。表达式写在以“<%=”开始、“%>”结束的标记中，其语法格式如下：

```
<%=  表达式 %>
```

下面是一个表达式的例子。

```
<font color="blue"><%= person.getName() %></font>
<b><%= new Date() %></b>.
```

这个表达式自动转化为字符串类型，然后插入表达式所在的 JSP 文件位置，还可以将表达式插入任何一行文本中。使用 JSP 的表达式时，必须注意以下事项：

- 表达式最后不能使用“;”结束。
- 表达式元素可以是任何一个符合 Java 标准的有效表达式。
- 表达式的属性可以作为其他 JSP 元素的属性值。

26.2.4　脚本集

在 JSP 中包含一段有效的 Java 程序代码。Scriptlet 使用语法如下：

```
<% code fragment %>
```

下面是一个脚本集的例子。

【范例 26-2】示例代码 26-2 是一个脚本集示例。

示例代码 26-2

```
<%
    Locale locale = request.getLocale();                    // 用于获取用户的本地化信息
    Calendar calendar = Calendar.getInstance(locale);
                                                            // 用于获取用户所在地的时间
    int hour = calendar.get(Calendar.HOUR_OF_DAY);          // 用于获取小时
    String greeting = "";
    if (hour <= 6) {
        greeting = "现在是凌晨。";                          //判断并设置输出语句
    } else if (hour <= 9) {
        greeting = "早上好！";                              //判断并设置输出语句
    } else if (hour <= 12) {
        greeting = "上午好！";                              //判断并设置输出语句
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
12        } else if (hour <= 18) {
13            greeting = "下午好! ";                    //判断并设置输出语句
14        } else if (hour <= 24) {
15            greeting = "晚上好! ";                    //判断并设置输出语句
16        } else {
17        }
18    %>
19
20    <html>
21    <head>
22    <!—设置页面-->
23        <title>欢迎页面</title>
24        <meta http-equiv="pragma" content="no-cache">
25        <meta http-equiv="cache-control" content="no-cache">
26        <meta http-equiv="expires" content="0">
27        <meta http-equiv="keywords" content="keyword1,keyword2,keyword3">
28        <meta http-equiv="description" content="This is my page">
29    </head>
30    <body>
31        <table>
32            <tr>
33                <td><%= greeting %></td>
34            </tr>
35        </table>
36    </body>
37    </html>
```

【运行结果】程序运行结果如图 26-10 所示。

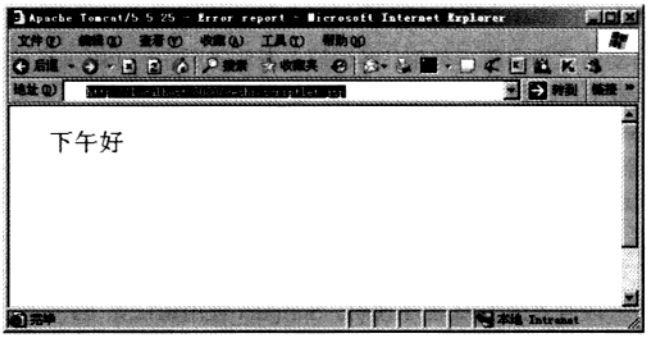


图 26-10　脚本运行效果

【代码解析】以上程序中，首先定义获取本地化的信息以及本地的时间（第 02、03 行）。定义字符串类型 hour 作为时间，并对 hour 进行判断，分别输出不同的内容（第 06～16 行）。最后调用字符串 getting，输出相应的内容（第 33 行）。

JSP 代码中可以将 Java 代码放在“<%”和“%>”之间，并且一个 Scriptlet 代码块可以包括多个 JSP 语句、表达式和方法等。使用 Scriptlet 必须注意以下事项。

- 声明变量和方法。
- 使用任何隐含对象，如 request、session、application 等，也可以使用<jsp:useBean>已经声明过的对象。
- 其他任何文本、HTML 标记，JSP 元素必须在 Scriptlet 外面。
- 当 JSP 页面被用户请求时，会执行 Scriptlet 代码，如果有显示内容，会存放于 out 对象中。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



26.3 JSP 指令元素

JSP 指令用于设置与整个 JSP 页面相关的属性。指令能够让 JSP 引擎按照指定的参数处理随后的 JSP 代码，相当于为 JSP 引擎设置全局变量。JSP 指令主要包括 page、include、taglib 三种。

26.3.1 Include 指令

Include 指令用于将文件插入到 JSP 页面中，这些文件可以是文本文件、HTML 文件或者是 JSP 文件。其基本语法格式如下：

```
<%@ include file="relativeURL" %>
```

其中，relativeURL 是该程序内另外一个 JSP 或者 HTML 文件的路径。下面以一个示例来说明，此例分为三个 JSP 文件，分别为 head.jsp、foot.jsp、include.jsp。

【范例 26-3】示例代码 26-3 是一个 include 指令示例。

示例代码 26-3

```
<%@ page language="java" contentType="text/html; charset=utf-8"%>
<html>
<head>
<title>JSP Scriptlets</title>
<link rel='stylesheet' type='text/css' href='css/style.css'>
</head>
<body>
<table width="100%" cellspacing=1 bgcolor=#cccc>
    <tr>
        <td bgcolor=#666666 colspan=7
            style="color: #ffffff; font-size:40px; height:60px; ">
                Include 指令示例</td>
    </tr>
    <tr>
        <%--设置头部的按钮连接“首页”--%>
        <td bgcolor=#cccc align=center>首页</td>
        <%--设置头部的按钮连接“新闻” --%>
        <td bgcolor=#cccc align=center>新闻</td>
        <%--设置头部的按钮连接“娱乐” --%>
        <td bgcolor=#cccc align=center>娱乐</td>
        <%--设置头部的按钮连接“博客” --%>
        <td bgcolor=#cccc align=center>博客</td>
    </tr>
</table>
  </body>
</html>
```

【代码解析】以上代码作为完整页面的一部分，用于显示页面的头部信息及一些功能连接（第 14～21 行）。

foot.jsp

```
<%@ page language="java" contentType="text/html; charset=utf-8"%>
<html>
<body>

<table width="100%" cellspacing=1 bgcolor=#cccc>
    <tr>
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
07          <%--设置页面页脚信息--%>
08      <td align=center bgcolor=#666666 style="color: #ffffff; font-size:14px;
        height:20px; ">
09              Copyright 2010-2012 &copyxuanyuanke
10          </td>
11      </tr>
12      <tr></tr>
13  </table>
14  </body>
15  </html>
```

【代码解析】以上代码用于显示底部信息，在页面上显示版本、版权信息（第 05～09 行）。

include.jsp

```
01  <%@ page language="java" contentType="text/html; charset=utf-8"%>
02  <%@ include file="head.jsp" %>
03  <html>
04  <!—设置页面显示信息 -->
05  <p style="line-height:22px; text-indent: 2em;">
06  <p style="line-height:22px; text-indent: 2em;">
07  <!—以下为设置的页面内容 -->这是一个 include 指令的示例<br/>
08
09  主要分为头部信息<br/>
10
11  和底部信息<br/>
12  头部为 head.jsp <br/>
13  底部为 foot.jsp <br/>
14
15  <p style="line-height:22px; text-indent: 2em;">
16  <%@ include file="foot.jsp" %>
17  </html>
```

【运行结果】程序运行结果如图 26-11 所示。



图 26-11　include 指令运行效果

【代码解析】以上代码是输出页面中的内容，并使用 include 指令将 head.jsp 和 foot.jsp 页面连接起来，作为一个完整的页面（第 02 行和第 16 行）。然后输出页面内容信息（第 09～13 行）。

26.3.2　Page 指令

page 指令用来定义页面的很多属性，例如脚本语言、编码方式以及导入的 jar 包等。一个

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

JSP 页面中可以包含有多个指令，除了 import 之外，其他的页面属性只允许出现一次，而且 page 指令可以放置在 JSP 页面的任何地方。指定的语法格式如下：

```
<%@page  atrribute = "value".....%>
```

以下是 page 指令属性 atrribute 的值，其中包括 language、import、session、buffer、autoFlush、isThreadSafe、info、errorPage、contentType、isErrorPage。

```
<%@ page
[ language="java" ]
[ import="{package.class / package.*}, ..." ]
[ session="true / false" ]
[ buffer="none / 8kb / sizekb" ]
[ autoFlush="true / false" ]
[ isThreadSafe="true / false" ]
[ info="text" ]
[ errorPage="relativeURL" ]
[ contentType="mimeType [ ;charset=characterSet ]" / "text/html ; charset=ISO-8859-1" ]
[ isErrorPage="true / false" ]
%>
```

页面属性具体描述如下。

- language：指定 JSP 页面使用的脚本语言。
- session：用于指定 JSP 是否使用 session，默认值为 true，表示 session 变量是可以被访问的；如果值为 false，则表示 session 变量值是不能被使用的。
- buffer：用于指定 JSP 网页的缓冲区大小，默认值是由服务器确定的。
- autoFlush ：作用是指定缓存区工作策略，如果值为 true，则会将数据传输到客户端；如果值为 false，则会出现异常。
- isThreadSafe：指定该 JSP 文件是否可以使用多线程，默认值为 true，可以同时处理多个用户的请求；如果值为 false，则只能处理一个请求。
- errorPage：用于指定出现异常时所调用的页面；也可以通过预定义的 exception 变量来指定错误页面。
- isErrorPage：指定该 JSP 文件是否显示错误页面。默认值为 false。
- contentType ：用于指定 JSP 文件的文件格式与字符集。



注意：errorPage 是用来配置本 JSP 网页的错误处理界面，而 web.xml 可以指定整个 Web 应用的错误处理界面。

26.3.3 Taglib 指令

使用 Taglib 指令可以使 JSP 页面支持 JSP 标签，而且还允许页面使用用户自定义标签先定义自己的标签库，然后为标签库编写配置文件，最后在 JSP 页面中使用该自定义标签。这样使用标签，就增加了代码的重用程度。用户可以把一些需要迭代显示的内容制作成一个标签。使用标签的语法格式如下：

```
<%@ taglib uri="URIToTagLibrary" prefix="tagPrefix" %>
```

其中的属性值 URIToTagLibrary 表示标签的描述符，也就是 JSP 的引擎自动根据其中所制定的内容去查找自定义标签库和描述文件，而属性 prefix 则是设置 JSP 文件中所引用标签的前缀，但是不能为 jsp、jspx、java、javax、servlet 等。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

26.4 JSP 动作元素

JSP 动作元素的主要作用就是根据其制定的动作进行相应的处理。利用 JSP 动作元素可以实现引入文件、转发请求到一个新的页面、重用 JavaBean 等功能。其中的动作主要包括：<jsp:forward>、<jsp:getProperty>、<jsp:include>、<jsp:plugin>、<jsp:setProperty>、<jsp:useBean>。

26.4.1 <jsp:forward>

forward 动作就是用来将请求信息发送到另一个页面中，其中只有一个 page 属性。forward 动作可以是 JSP 之间的转发，也可以是 Servlet 或者静态资源文件之间的请求转发。其语法格式如下：

```
<jsp:forward page={"relativeURL" / "<%= expression %>"} />
```

或者

```
<jsp:forward page={"relativeURL" / "<%= expression %>"} >
<jsp:param name="parameterName"
value="{parameterValue / <%= expression %>}" />
</jsp:forward>
```

例如：

```
<jsp:forward page="/servlet/login" />
<jsp:forward page="/servlet/login">
<jsp:param name="usr" value="jack" />
<jsp:param name="password" value="654321" />
</jsp:forward>
```

其中的 page 值必须为一个表达式或者是一个字符串，它用来指定文件的 URL，这个文件可以是一个 JSP 文件或者是程序代码段。

jsp:forward 中有属性 name 和 value，其中 name 用来指定参数名，value 则用来指定参数的值，参数可以是一个或者多个，但是必须要发给一个动态的文件，该动作元素使用最广泛的就是登录，通过验证后跳转到主页面中。

26.4.2 <jsp:useBean>

useBean 动作用来装载一个在 JSP 页面中使用的 JavaBean，JavaBean 是用 Java 语言编写的可重用的软件组件。该动作需要制定 JavaBean 的作用范围，其语法格式如下：

```
<jsp:useBean id="id" scope="scope" class=" className " type="teyeName"/>
```

- id：该变量确定在指定范围内的 Bean 变量的唯一值。可以通过 id 值来对 JavaBean 的对象实例进行引用。
- class：指定 Bean 的路径名或者是类名。该属性对大小写比较敏感，而且必须有一个公用的无参构造，不可以为抽象类。
- typeName：type 可以是本身、父类或者是接口，如果没使用 class 来指定 BeanName，JavaBean 则不会被实例化。
- scop：用来指定 JavaBean 变量存在的有效范围。scope 的值可以是“page”、“request”、“session”、“application”中的任意一个。

注意：如果要创建 session 范围的 Bean 实例，必须在<%@ page%>的指令中指定 session = “true”，否则会发生错误。另外，不能同时使用 class 属性和 BeanName 属性。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



【范例 26-4】示例代码 26-4 是一个 useBean 示例。

示例代码 26-4

```
01  public class UseBean {
02       //获得当前日期
03       Date date =new Date();
04        public String today(String s)
05       {
06            //初始化日期
07          SimpleDateFormat formatter = new SimpleDateFormat(s);
08          return formatter.format(date);
09       }
10  }
```

【代码解析】以上代码中定义了 Date 类型的 date 属性，并实现方法 today，用于返回本地的时间信息（第 03～08 行）。

useBean.jsp

```
01  <%@ page language="java" import="java.util.*" pageEncoding="utf-8"%>
02  <%
03  String path = request.getContextPath();
04  String basePath = request.getScheme()+"://"+request.getServerName ()+":"+
request.getServerPort()+path+"/";
05  %>
06
07  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
08  <html>
09    <head>
10    </head>
11
12    <body>
13      <jsp:useBean id="firstBean" class="cn.com.ch26.UseBean" scope="page" />
14        现在是北京时间：<br>
15        <%=firstBean.today("yyyy-MM-dd HH:mm:ss")%>
16
17    </body>
18  </html>
```

【运行结果】程序运行结果如图 26-12 所示。



图 26-12　useBean.jsp 运行效果

【代码解析】以上代码中，首先调用 UseBean 类中的 today 方法，设置本地时间的输出格式，并打印显示（第 13～15 行）。



注意：访问 Bean 实例通常有两种方法，一种是通过 jsp:getProperty 方法获取属性值，另一种是通过 Bean 的实例名调用 getter 方法直接获取属性值。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

26.4.3 <jsp:setProperty>

通过对<jsp:useBean>的学习，知道 useBean 可以获得 JavaBean 实例，如果需要修改 JavaBean 的属性，可以用<jsp:setProperty>动作元素来实现。<jsp:setProperty>动作有以下几个属性。

- name：该属性是不可缺少的。通过这个属性找到对应的 JavaBean 实例，并赋值。
- property：该属性是不可缺少的。通过它找到 JavaBean 实例中具体的属性进行赋值。如果把 property 属性值设置为“*”，就可以把所有与 JavaBean 属性名匹配的请求参数传递给 JavaBean 中对应属性的 set 方法。
- value：该属性是可选属性。用来指定 JavaBean 属性的值。
- param：该属性是可选属性。它和 value 属性不能同时使用，它用来指定某个参数作为 JavaBean 属性的值。

<jsp:setProperty>有两种用法：在<jsp:useBean>和</jsp:useBean>之间插入<jsp:setProperty>动作。这种情况只有在创建了 JavaBean 实例以后，<jsp:setProperty>动作才会被执行。在 JSP 中的写法如下：

```
<jsp:useBean id="" beanName="" scope="page" type="">
    <jsp:setProperty name="" property="" value=""/>
    <jsp:setProperty name="" property="" param=""/>
</jsp:useBean>
```

在<jsp:useBean>动作外面使用<jsp:setProperty>动作。这种情况下，不论 JavaBean 实例是否被创建，<jsp:setProperty>动作都会被执行。在 JSP 中的写法如下：

```
<jsp:useBean id="" beanName="" scope="page" type=""/>
<jsp:setProperty name="" property="" param=""/>
<jsp:setProperty name="" property="" value=""/>
```

对<jsp:setProperty>有所了解以后，下面通过一个例子来加深对它的认识。新建一个 SetProperty 类，用于记录 name 和 say 的值。

【范例 26-5】示例代码 26-5 是一个 setProperty 示例。

示例代码 26-5

```
01  public class SetProperty {
02      private String name;
03      private String say;
04      public String getName() {
05          return name;
06      }
07      public void setName(String name) {
08          this.name = name;
09      }
10      public String getSay() {
11          return say;
12      }
13      public void setSay(String say) {
14          this.say = say;
15      }
16  }
```

【代码解析】以上代码中，首先声明了 name、say 属性（第 02、03 行），在该类中实现属性所对应的 set/get 方法（第 04～15 行），

setProperty.jsp

```
01  <%@ page language="java" import="java.util.*" pageEncoding="utf-8"%>
02  <%
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
03  String path = request.getContextPath();
04  String basePath = request.getScheme()+"://"+request.getServerName()+ ":"+
    request.getServerPort()+path+"/";
05  %>
06  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
07  <html>
08    <head>
09     </head>
10   <body>
11           <h1>使用 setProperty 的例子</h1>
12      <p>
13      <br>
14      <h2>在 jsp:useBean 中的 jsp:setProperty</h2>
15      <jsp:useBean id="test" class="cn.com.ch26.SetProperty">
16        <jsp:setProperty name="test" property="name" value="名字: setProperty"/>
17        <jsp:getProperty property="name" name="test"/>
18      </jsp:useBean>
19      <p>
20      <hr>
21      <h2>在 jsp:useBean 外的 jsp:setProperty</h2>
22      <jsp:setProperty name="test" property="say" value="我是 setProperty"/>
23      <jsp:getProperty name="test" property="say"/>
24    </body>
25  </html>
```

【运行结果】程序运行结果如图 26-13 所示。

【代码解析】以上代码中使用 jsp:usebean 标签连接实体类（第 15 行），然后使用 setProperty 标签设置 setProperty 并显示输出（第 16 行、第 17 行、第 22 行、第 23 行）。

上述 JSP 代码中，一个<jsp:setProperty>动作在<jsp:useBean>和</jsp:useBean>之间，它通过 value="名字：setProperty"给 JavaBean 中的 name 属性赋值；还有一个<jsp:setProperty>动作在<jsp:useBean>外面，它通过 value="我叫 setProperty"给 JavaBean 中的 say 属性赋值。

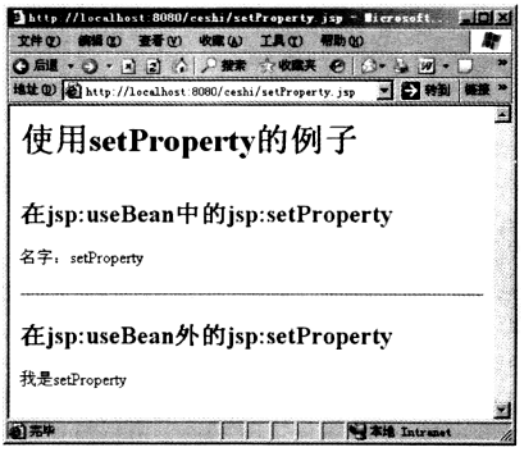


图 26-13 使用 setProperty 的运行效果

26.4.4 <jsp:getProperty>

jsp:getProperty 指令操作通常与 jsp:useBean 操作配合使用，以获取 Bean 的属性值，并显示在页面中。它访问的属性会自动转化为 String 类型，然后发送给输出流。jsp:getProperty 的使

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

用格式如下：

```
<jsp:getProperty name="BeanName" property="propertyName" />
```

例如：

```
<jsp:setProperty name="user" property="*"/>
<jsp:getProperty name="user" property="name"/>
<jsp:getProperty name="user" property="age"/>
```

- name="BeanName"：name 用指定 Bean 的名字，是必须有的属性。
- property="propertyName"：property 用于指定 Bean 的属性名。



注意：不能使用 jsp:getProperty 获取一个已经被索引的属性。

26.4.5 <jsp:include>

jsp:include 指令用于请求过程中，在 JSP 页面里包含静态或动态文本。被包含的对象只能具有 JspWriter 对象的访问权限，并且不能设置头或者 Cookies。jsp:include 指令的运行效率比<%@include file="relativeURL"%>低。jsp:include 指令使用格式如下：

```
<jsp:include page="{relativeURL / <%= expression%>}" flush="true" />
```

或者

```
<jsp:include page="{relativeURL / <%= expression %>}" flush="true" >
<jsp:param name="parameterName" value="{parameterValue / <%= expression %>}" />
</jsp:include>
```

例如：

```
01  <%@ page language="java" %>
02  <html>
03  <head><title> Include </title></head>
04  <body>
05  <jsp:include page=="head.jsp" %>
06  <jsp:include page ="body.jsp" %>
07  <jsp:include page ="footer.jsp" %>
08  </body>
09  </html>
```

- page：该参数是设置一个相对路径，或者是返回一个相对路径的表达式。路径以“/”开始，表示该路径是上下文关系路径。如果以文件名或者路径开始，表示该类位于当前 JSP 文件所在的目录。
- flush：该属性中的参数必须为 true，不能设置为 false。默认值为 false。
- <jsp:param name="parameterName" value="{parameterValue / <%= expression %>}" />：jsp:param 用来向动态页面传递参数。jsp:include 允许包含动态或者静态文件，但结果不同，如果是静态文件，直接将包含的文件添加到 JSP 文件中；如果是动态文件，这个文件将会被 JSP 编译器编译，并且可以通过 jsp:param 传递参数。



注意：jsp:include 包含的是动态的内容，即包含的内容可以改变，在执行时才能确定；而<%@ include %>方法包含的文件是静态的，即一经编译，就不会再改变。

26.4.6 <jsp:plugin>

jsp:plugin 用来产生特别的标识，如 Object、Embed 等，它可以装入 Applet 或者 JavaBean。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



当 JSP 文件被编译之后发送给浏览器，浏览器就可根据不同的版本替换成<object>或者<embed>元素。通常会指定是 Applet、Bean 或类名等。jsp:plugin 使用格式如下：

```
<jsp:plugin
type="bean / applet"
code="classFileName"
codebase="classFileDirectoryName"
[ name="instanceName" ]
[ archive="URIToArchive, ..." ]
[ align="bottom / top / middle / left / right" ]
[ height="displayPixels" ]
[ width="displayPixels" ]
[ hspace="leftRightPixels" ]
[ vspace="topBottomPixels" ]
[ jreversion="JREVersionNumber / 26.1" ]
[ nspluginurl="URLToPlugin" ]
[ iepluginurl="URLToPlugin" ] >
[ <jsp:params>
[ <jsp:param name="parameterName" value="{parameterValue / <%= expression %>}" /> ]
</jsp:params> ]
[ <jsp:fallback> text message for user </jsp:fallback> ]
</jsp:plugin>
```

该动作元素中的属性如表 26-1 所示。

表 26-1　jsp:plugin 属性

属　性	说　明
codebase	用来设置目录，这个目录是将被执行的 Java 类所在的目录。不设置这个属性的情况下，默认使用<jsp:plugin>动作的 JSP 所在目录
code	code 属性值是一个将被插件执行的 Java 类名称，这个类名以“.class”结尾。这个被执行的 Java 类必须在 codebase 指定的目录下
type	用来设置被执行对象的类型，没有默认值。开发人员必须要指定被执行对象是 Bean 还是 Applet。这个属性的值只有“Bean”和“Applet”两个
align	设置被执行对象的位置，它的值有：bottom、top、middle、left、right
archive	它的值是一些用逗号“,”分开的路径名，用来预先加载将要执行的类
Height	设置被执行对象（Applet 或者 Bean）高度的值，它的单位是像素
hspace	它的值也是以像素为单位，用来设置 Applet 或者 Bean 执行时，在屏幕中留下的空间
iepluginurl	其值是一个标准的 URL 地址，这个地址是 IE 用户能够使用的 JRE 下载地址
nspluginurl	其值也是一个标准的 URL 地址，这个地址是 Netscape Navigator 用户能够使用的 JRE 下载地址
jreversion	用来表示执行对象（Applet 或者 Bean）在执行时所需要的 JRE 版本，默认值是 1.2
name	用来表示执行对象（Applet 或者 Bean）的名称
width	用来设置显示对象（Applet 或者 Bean）宽度的值，以像素为单位



26.5　内置对象

为了方便界面的开发，JSP 为用户提供了一些内置对象，这些内置对象不需要开发者实例化，在所有的 JSP 页面都能直接使用内置对象。JSP 内置对象包括：request、response、session、out、application、pageContext、exception、config、page，下面分别介绍这些内置对象的使用。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

26.5.1 out 对象

out 对象是向客户端输出内容时比较常用的对象，该对象通过 PrintWriter 类在服务器向客户端发送信息，并且把信息输出到客户端。out 对象常用的方法如表 26-2 所示。

表 26-2 out 对象方法

方 法	说 明
print(type)、println(type)	该方法用于输出 type 类型的值，后者还输出一个换行
newLine()	该方法用于输出一个换行字符
flush()	该方法用于输出缓冲区里的数据
close()	该方法用于关闭输出流
clearBuffer()	该方法用于清除缓冲区里的数据，并输出到客户端
clear()	该方法用于清除缓冲区里的数据，但并不输出到客户端
getBufferSize()	该方法用于获得缓冲区的大小
getRemaining()	该方法用于获得缓冲区闲置空间的大小
isAutoFlush()	该方法用于返回 AutoFlush 的布尔值

对于服务器内容的输出，默认是首先将输出内容写入客户端缓存区，但是也有以下三种特殊情况会将缓存区的内容输出到客户端界面中。

- JSP 页面的内容输出完毕。
- 指定的缓存区已满。
- 强制 JSP 中调用了 out.flush()或者 response.flushbuffer()。

26.5.2 request 对象

request 对象是与用户请求相关的对象，该对象包含在客户端与服务器之间所要传送的数据，例如，所有的客户端对服务器的请求信息来源和头信息、Cookie 等与请求相关的参数值。HttpServletRequest 定义了访问参数、对象属性、客户端与 Servlet 之间的通信协议。request 对象中的常用方法如表 26-3 所示。

表 26-3 request 对象方法

方 法	说 明
isUserInRole(String role)	该方法的作用是判断认证用户是否属于 role 中的成员
getAttribute(String name)	该方法的作用是返回指定的属性值，若属性不存在，则返回 null
getAtrributeNames()	该方法的作用是返回 request 所有属性的名字集合
getCookies()	该方法的作用是返回客户端的所有 Cookie 对象数组
getCharacterEncoding()	该方法的作用是返回 request 的字符编码方式
getContentLength()	该方法的作用是返回 request 的 Body 长度，若不确定长度，则返回-1
getHeader(String name)	该方法的作用是返回 HTTP 协议的文件头信息
getHeaders(String name)	该方法的作用是返回所有 request Header 的值
getHeaderNames()	该方法的作用是返回所有 request Header 的名字
getInputStream()	该方法的作用是返回请求的输入流
getMethod()	该方法的作用是返回客户端向服务器端传递数据的方式，如 get、post 等
getParameter(String name)	该方法的作用是返回客户端传递到服务器表单的参数值

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

续表

方　法	说　明
getParameterNames()	该方法的作用是返回客户端传递到服务器表单的所有参数名字
getParameterValues(String name)	该方法的作用是返回客户端传递到服务器表单中指定参数的所有值
getProtocol()	该方法的作用是返回客户端向服务器端传递数据的协议名称
getQueryString()	该方法的作用是返回客户端向服务器端以 Get 方法传送的查询字符串
getRequestURI()	该方法的作用是返回发出请求字符串的客户端 URI
getRomteAddr()	该方法的作用是返回客户端的主机 IP 地址
getRomteHost()	该方法的作用是返回客户端的主机名称
getSession([Boolean create])	该方法的作用是返回请求相关的 session。参数是可选的，若为 True，当客户端没有创建 session，就会创建一个 session
getServerName()	该方法的作用是返回服务器名
getServletPath()	该方法的作用是返回服务器端的脚本文件路径
getServerPort()	该方法的作用是获取服务器的端口号
removeAttribute(String name)	该方法的作用是删除 request 中的一个属性
setAttribute(String name,java.lang.Object objt)	该方法的作用是设置指定参数的值

下面由一个示例来进行说明，在 MyEclipse 中 Web 项目的 WebRoot 目录下创建文件“request.jsp”、“request.html”，在项目包下创建“Person.java”，然后在其中输入以下代码。

request.jsp 页面用于反馈用户输入的信息，其主要代码如下。

【范例 26-6】示例代码 26-6 是一个 request 对象示例。

示例代码 26-6

```
<%@ page language="java" contentType="text/html; charset=utf-8" %>
<html>
<head>
<title>Java Bean Actions</title>
<link rel='stylesheet' type='text/css' href='css/style.css'>
</head>
<body><br/>

<%-- 声明 Person 类对象 person --%>
<jsp:useBean id="person" class="cn.com.jsp.Person" scope="page" />

<%-- 设置 person 的所有属性，属性值从 request 中自动取得。*表示所有属性 --%>
<jsp:setProperty name="person" property="*" />

<div align="center">
    <form action="method.jsp" method="get">
            <legend>请填写用户信息</legend>
            <table align="center" width="400">
                <tr>
                    <td align="right" style="font-weight:bold; ">姓名：</td>
                    <td>
                        <%-- 获取 person 的 name 属性 --%>
                        <jsp:getProperty name="person" property="name" />
                    </td>
                </tr>
                <tr>
                    <td align="right" style="font-weight:bold; ">年龄：</td>
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
28                          <td>
29                              <%-- 获取 person 的 age 属性 --%>
30                              <jsp:getProperty name="person" property="age" />
31                          </td>
32                      </tr>
33                      <tr>
34                          <td align="right" style="font-weight:bold; ">性别: </td>
35                          <td>
36                              <%-- 获取 person 的 sex 属性 --%>
37                              <jsp:getProperty name="person" property="sex" />
38                          </td>
39                      </tr>
40                      <tr>
41                          <td align="right" style="font-weight:bold; "></td>
42                          <td>
43                          <input type="button" onclick="history.go(-1); " value="
                                    返回 " class="button">
44                          </td>
45                          </tr>
46                  </table>
47      </form>
48  </div>
49  </body>
50  </html>
```

【代码解析】以上程序中，首先声明 Person 类对象，并接收到用户在 request.html 页面中传入的用户信息（第 09～13 行），最后将用户信息打印输出到页面中（第 15～43 行）。

Person.java

```
01  public class Person {
02      private String name;                    // 姓名
03      private int age;                        // 年龄
04      private String sex;                     // 性别
05      private boolean secret;                 // 是否保密信息
06
07      public int getAge() {                   // age 的 getter 方法
08          return age;
09      }
10      public void setAge(int age) {           // age 的 setter 方法
11          this.age = age;
12      }
13      public String getName() {               // name 的 getter 方法
14          return name;
15      }
16      public void setName(String name) {  // name 的 setter 方法
17          this.name = name;
18      }
19      public String getSex() {                // sex 的 getter 方法
20          return sex;
21      }
22      public void setSex(String sex) {                // sex 的 setter 方法
23          this.sex = sex;
24      }
25      public void setSecret(boolean secret) {     // secret 的 getter 方法
26          this.secret = secret;
27      }
28      public boolean isSecret(boolean secret) {
29          return secret;
30      }
31  }
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

【代码解析】以上代码中，首先声明姓名、年龄、性别以及保密属性（第 02～05 行），并依次设置属性的 set/get 方法（第 07～27 行）。

request.html

```
01  <html>
02  <head>
03  <title>Java Bean Actions</title>
04  <link rel='stylesheet' type='text/css' href='css/style.css'>
05  </head>
06  <body><br/>
07
08  <div align="center">
09      <form action="request.jsp" method="post">
10          <legend>请填写用户信息</legend>
11          <table align="center" width="400">
12              <tr>
13              <!--设置姓名-->
14              <td align="right" style="font-weight:bold; ">姓名：</td>
15              <td><input type="text" name="name" value="" style="width:200px;
                    "/></td>
16              </tr>
17              <tr>
18                  <!--设置年龄-->
19              <td align="right" style="font-weight:bold; ">年龄：</td>
20              <td><input type="text" name="age" value="" style="width:200px;
                    "/></td>
21                  </tr>
22                  <tr>
23                  <!--设置性别-->
24              <td align="right" style="font-weight:bold; ">性别：</td>
25                      <td>
26                  <input type="radio" name="sex" value="Male" />Male
27                  <input type="radio" name="sex" value="Female" />Female
28                      </td>
29                  </tr>
30                  <tr>
31              <td align="right" style="font-weight:bold; "></td>
32                  <td>
33          <!--设置按钮-->
34          <input type="submit" name="search" value=" 提交用户信息 " class="button">
35                      </td>
36                  </tr>
37
38              </table>
39      </form>
40  </div>
41  </body>
42  </html>
```

【运行结果】程序运行效果如图 26-14 所示。

【代码解析】以上代码中，首先设置一个登录界面，用于填写用户的信息（第 10～34 行），并将用户的信息传递到 request.jsp 页面（第 09 行）。在这个 JSP 程序中，并没有使用脚本，而是全部使用 JSP 行为完成，使用这种方式使 JSP 的代码显得更加整洁，有很好的可读性，建议在以后的开发中都使用这种方式。

26.5.3 response 对象

response 对象的作用是对客户端的 request 做出响应，向客户端输出信息。response 对象是

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

与 request 相对应的一个对象。

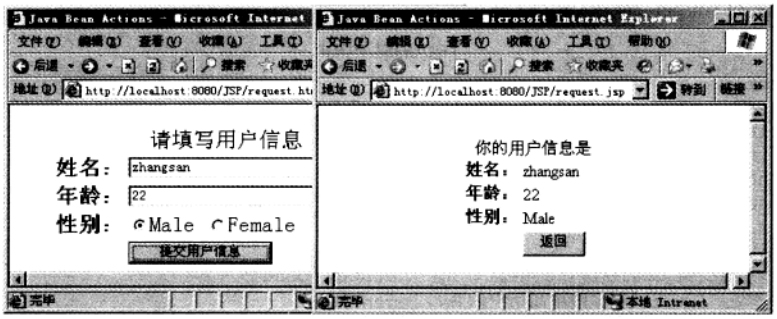


图 26-14　request 运行效果

对象 response 是 javax.servlet.ServletResponse 类的实例，代表客户端请求的响应。服务器端的任何输出都通过 response 对象发送到客户端浏览器，每次服务器端都会响应一个 response 实例。response 隐藏对象常用到的方法如表 26-4 所示。

表 26-4　request 对象方法

方　法	说　明
addCookies(Cookies cookies)	该方法的作用是 response 添加一个 Cookies 对象来保存客户端用户信息
addHeader(String name,String value)	该方法的作用是添加 HTTP 文件头信息，如果头信息已经存在，就覆盖原头信息
containsHeader(String name)	该方法的作用是返回 HTTP 文件头是否存在指定名字的头信息
encodeURL()	该方法的作用是通过 SessionId 来封装 URL
flushBuffer()	该方法的作用是将当前缓冲区数据强制输出
getBufferSize()	该方法的作用是返回缓冲区的大小
getOutputStream()	该方法的作用是返回客户端的输出流对象
sendError(int)	该方法的作用是向客户端发送错误信息
sendRedirect(String localtion)	该方法的作用是将请求转发到另外一个位置
setContentType(String contentType)	该方法的作用是设置相应的 MIME 类型
setHeader(String name,String value)	该方法的作用是设置 HTTP 头文件的值，如果该名字的值已经存在，覆盖旧值

以下示例是判断用户登录，如果获取到用户输入的用户名和密码都不为空，则跳转到登录成功界面，否则弹出登录失败界面。其中主要有 login.jsp 页面和 welcome.jsp 页面，login.jsp 页面是设置用户的登录界面。

【范例 26-7】示例代码 26-7 是一个 useBean 示例。

示例代码 26-7

```
<%@ page language="java" pageEncoding="utf-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <body>
    <form action="redirect.jsp">
          <table border="0">
              <tr>
                  <td align="left">用户名：</td>
                  <td align="left"><input type="text" name="username"></td>
              <tr>
                  <td align="left">密　码：</td>
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
12                  <td align="left"><input type="password"
                                    name="password"></td>
13              </tr>
14          </table>
15          <input type="submit" value="登录">
16          <input type="reset" value="取消">
17      </form>
18    </body>
19  </html>
```

【代码解析】以上代码用于获取用户所输入的用户名及密码信息（第 06～16 行），然后将获得的用户名和密码信息传递到 redirect.jsp 页面（第 05 行）。

redirect.jsp

```
01  <%@ page language="java" pageEncoding="utf-8"%>
02  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
03  <html>
04    <body>
05     <%
06  String username = (String)request.getParameter("username");//获取登录的用户名
07  String password = (String)request.getParameter("password");//获取登录的密码
08          if(username != null && password != null){    //判断登录的用户名是否为空
09              if(username.equals(username)){           //用户登录认证过程
10              response.sendRedirect("welcome.jsp");    //登录成功，重定向到欢迎页面
11              }else{
12              response.sendRedirect("error.jsp");      //登录错误，重定向到错误页面
13              }
14          }
15      %>
16    </body>
17  </html>
```

【代码解析】以上代码中，首先获取到用户的登录名和密码（第 06、07 行），并根据用户输入的信息判断是否也可跳转到欢迎界面（第 08～12 行）。

welcome.jsp

```
01  <%@ page language="java" import="java.util.*" pageEncoding="utf-8"%>
02  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
03  <html>
04    <head>
05    </head>
06      <body>
07      欢迎访问本网站！ <br>
08    </body>
09  </html>
```

【运行结果】程序运行结果如图 26-15 和图 26-16 所示。

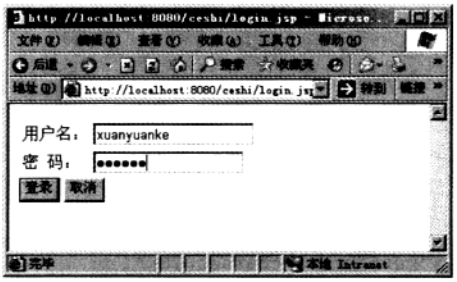


图 26-15 登录框效果



图 26-16 登录成功效果

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

【代码解析】以上代码中，只包含有“欢迎访问本网站”（第 07 行），是在登录成功后所跳转的页面。

26.5.4 session 对象

session 是 javax.servlet.http.HttpSession 类的实例。session 与 cookie 是记录客户访问信息的两种机制，session 用于在服务器端保存用户信息，cookie 用于在客户端保存用户信息。Servlet 中需要通过 request.getSession()来获取 session 对象，而 JSP 中可以直接使用。如果 JSP 中配置了<%@ page session="false"%>，则隐藏对象 session 不可用。session 中的方法如表 26-5 所示。

表 26-5 session 方法

方 法 名	说 明
getAttribute(String name)	该方法的作用是返回 session 对象中指定名字的属性
getAttributeNames()	该方法的作用是返回 session 对象中的属性名字
getCreateionTime()	该方法的作用是返回 session 对象创建时间，可以精确到千分之一秒
getId()	该方法的作用是返回 session 对象唯一的标识
getLastAccessedTime()	该方法的作用是返回 session 对象所属客户端最后请求的时间，可以精确到千分之一秒
getMaxInactiveInterval()	该方法的作用是返回 session 对象的生命时间(秒)，负值表示已经过期
invalidate()	该方法的作用是销毁 session
isNew()	该方法的作用是判断是否是一个新创建的 session 对象。当用户访问支持 session 的 JSP 页面时，它将被创建
removeAttribute(String name)	该方法的作用是删除 session 对象中指定的属性
setAttribute(String name,Object value)	该方法的作用是为 session 对象设置指定属性的值

下面是一个 session 的示例，用于实现对访问次数的统计，包括 Number.java、session1.jsp 和 session2.jsp。其中 Number.java 是一个实体类，用于记录访问的次数。

【范例 26-8】示例代码 26-8 是一个 session 对象示例。

示例代码 26-8

```
01  public class Number {
02      private int num = 0;                    //计数属性
03
04      public int getNum() {
05          num++;                              //每次访问时，自动加 1
06          return num;
07      }
08
09      public void setNum(int num) {           //设置 num 的 set 方法
10          this.num = num;
11      }
12
13  }
```

【代码解析】以上代码中，首先声明了整型的 num，用于记录登录次数（第 02 行）。然后设置 get/set 方法，并在 get 方法中设置 num 的值自动累加（第 04～10 行）。

session1.jsp

```
01  <%@ page language="java" contentType="text/html; charset=GBK"
02      pageEncoding="GBK"%>
03  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=GBK">
<title>Insert title here</title>
</head>
<%-- 设置颜色 --%>
<body bgcolor="#FFFFFF">
<%-- 设置JavaBean的地址--%>
<jsp:useBean id="visitcount" class="cn.com.ch26.Number" scope="session"/>
<br>
<p><font color="blue">
<%-- 获得数值并输出 --%>
<h1>第<jsp:getProperty name="visitcount" property="num"/>次访问！</h1>
</font></p>
<a href="session2.jsp">下一个页面</a>
</body>
</html>
```

【代码解析】以上代码中，首先设置 JavaBean 的地址，获取到 num 的值后输出（第 13～17 行）。设置一个连接，指向 session2.jsp 页面（第 19 行）。

session2.jsp

```
<%@ page language="java" contentType="text/html; charset=GBK"
    pageEncoding="GBK"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<%-- 页面设置 --%>
<meta http-equiv="Content-Type" content="text/html; charset=GBK">
<title>Insert title here</title>
</head>
<%-- 设置颜色 --%>
<body bgcolor="#FFFFFF">
<br>
<p><font color="blue">
<%-- 获得数值并输出 --%>
<h1>第<jsp:getProperty name="visitcount" property="num"/>次访问!!</h1>
</font></p>
</body>
</html>
```

【运行结果】程序运行结果如图 26-17 所示。



图 26-17 session 对象效果

【代码解析】以上代码中，首先设置字体颜色，然后获取到 num 的值，并输出显示（第 12～

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

16 行)。

26.5.5 application 对象

application 对象是 javax.servlet.ServletContext 接口的实现。application 封装了 JSP 所在的 Web 应用程序的信息，例如，web.xml 中配置的全局的初始化信息。Servlet 中 application 对象需要通过 ServletConfig.getServletContext()来获取。整个 Web 应用程序对应一个 application 对象。

application 对象与 session 对象的作用类似，最大的不同在于作用域，Web 服务器会为每个用户创建一个 session，但所有的用户会共享一个 application，一旦创建了 applicatin 对象，这个对象将一直存在，直到服务器关闭。application 对象最常被使用于存取环境的信息，因为环境的信息通常都储存在 ServletContext 中，所以常利用 application 对象来存取 ServletContext 中的信息。application 中的方法如表 26-6 所示。

表 26-6 application 方法

方 法 名	说 明
Object getAttribute(String name)	该方法的作用是返回给定名的属性值
Enumeration getAttributeNames()	该方法的作用是返回所有可用属性名
void setAttribute(String name,Object obj)	该方法的作用是设定属性的属性值
void removeAttribute(String name)	该方法的作用是删除 application 中的属性及其属性值
String getServerInfo()	该方法的作用是返回 JSP 或者 Servlet 的服务器引擎名、版本号及相关信息
String getRealPath(String path)	该方法的作用是返回服务器中虚拟路径的真实路径
ServletContext getContext(String uripath)	该方法的作用是返回该服务应用的 application 对象
int getMajorVersion()	该方法的作用是返回服务器支持的最大版本号
int getMinorVersion()	该方法的作用是返回服务器支持的最小版本号
String getMimeType(String file)	该方法的作用是返回指定文件的 MIME 类型
URL getResource(String path)	该方法的作用是返回指定文件及目录的 URL
InputStream getResourceAsStream(String path)	该方法的作用是返回指定文件及目录的输入流
RequestDispatcher getRequestDispatcher(String uripath)	该方法的作用是返回指定文件及目录的 RequestDispatcher 对象
Servlet getServlet(String name)	该方法的作用是返回指定名的 Servlet 对象
Enumeration getServlets()	该方法的作用是返回所有 Servlet 的枚举变量
Enumeration getServletNames()	该方法的作用是返回所有 Servlet 名字的枚举变量
void log(String msg)	该方法的作用是把指定消息写入 Servlet 的日志文件
void log(Exception exception,String msg)	该方法的作用是将指定异常及错误消息写入 Servlet 的日志文件
void log(String msg,Throwable throwable)	该方法的作用是将错误消息和 Throwable 异常写入 Servlet 的日志文件

下面也是一个记录访问次数的实例，和前面不同的是，该实例中的作用域不同。该实例中有 ApplicationTest.java、application1jsp 和 application2.jsp 三个文件。

【范例 26-9】示例代码 26-9 是一个 application 对象示例。

示例代码 26-9

```
01  public class ApplicationTest
02  {
03      private int num = 0;                          //计数属性
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
04
05      public int getNum() {
06          num++;                                    //每次访问时，自动加 1
07          return num;
08      }
09
10      public void setNum(int num) {                 //设置 num 的 set 方法
11          this.num = num;
12      }
13  }
```

【代码解析】以上代码中，首先声明 num 属性用于计算登录的次数（第 03 行）。然后设置 num 对应的 set/get 方法，并在 get 方法中设置 num 值自动累加（第 05～11 行）。

application1.jsp

```
01  <%@ page language="java" contentType="text/html; charset=GBK"
02     pageEncoding="GBK"%>
03  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
04  "http://www.w3.org/TR/html4/loose.dtd">
05  <%@page import="cn.com.jsp.ApplicationTest"%>
06  <html>
07  <head>
08  <%-- 页面设置 --%>
09  <meta http-equiv="Content-Type" content="text/html; charset=GBK">
10  <title>Insert title here</title>
11  </head>
12  <%-- 设置颜色 --%>
13  <body bgcolor="#FFFFFF">
14  <%-- 设置 JavaBean 的路径 --%>
15  <jsp:useBean id="visitcount" class="ApplicationTest" scope="application"/>
16  <br>
17  <p><font color="blue">
18  <%-- 获得数值并输出 --%>
19  <h1>第<jsp:getProperty name="visitcount" property="num"/>次访问！</h1>
20  </font></p>
21  </body>
22  </html>
```

【代码解析】以上代码中，首先设置显示字体的颜色，并设置获取 JavaBean 的路径，获得 num 值（第 09～15 行）。在获取到 num 的值后，在页面中显示用户所访问的次数（第 19 行）。

application2.jsp

```
01  <%@ page language="java" contentType="text/html; charset=GBK"
02     pageEncoding="GBK"%>
03  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
04  "http://www.w3.org/TR/html4/loose.dtd">
05  <html>
06  <head>
07  <meta http-equiv="Content-Type" content="text/html; charset=GBK">
08  <title>Insert title here</title>
09  </head>
10  <body bgcolor="#FFFFFF">
11  <br>
12  <p><font color="blue">
13  <%-- 获得数值并输出 --%>
14  <h1>第<jsp:getProperty name="visitcount" property="num"/>次访问!!</h1>
15  </font></p>
16  </body>
17  </html>
```

【运行结果】运行结果如图 26-18 所示。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



图 26-18　application 对象示例效果

【代码解析】以上代码中，首先设置显示字体的颜色，获得 num 值并在页面中显示用户所访问的次数（第 12～14 行）。JavaBean 被创建，则整个应用之内都会有效，也就是说，在启动服务器后，浏览 application1.jsp，显示“第 1 次访问”，再打开一个浏览器，访问 application2.jsp，则会显示“第 2 次访问”。

26.5.6　pageContext 对象

隐藏对象 pageContext 为 javax.servlet.jsp.PageContext 类的实例。pageContext 对象代表当前 JSP 页面编译后的内容，通过 pageContext 能够获取 JSP 中的资源。pageContext 的常用方法如表 26-7 所示。

表 26-7　pageContext 方法

方 法 名	说　明
foward(String relativeUrlPath)	该方法的作用是将页面重定向到另外的资源(包括页面和 Servlet 组件)
getAttribute(String name[,int scope])	该方法的作用是返回指定范围 scope 的指定参数的值
getException()	该方法的作用是返回当前的 Exception 对象
getRequest()	该方法的作用是返回当前的 Request 对象
getResponse()	该方法的作用是返回当前的 Response 对象
getServletConfig()	该方法的作用是返回当前页面的 ServletConfig 对象
getServletContext()	该方法的作用是返回 ServletContext 对象
getSession()	该方法的作用是返回当前页面的 Session 对象
findAttribute()	该方法的作用是完成按照页面、请求、会话，以及应用程序范围的顺序对属性的搜索
setAttribute()	该方法的作用是用来设置特定范围的属性值，默认为页面范围
removeAttribute()	该方法的作用是删除特定范围指定的属性值，默认为页面范围

26.5.7　exception 对象

exception 对象只存在于错误的页面，该对象是 java.long.Throwable 的一个实例。如果设置有错误页面，必须将 isErrorPage 属性设置为 true，否则就会因无法编译而报错。以下是 exception 的常用方法。

- String getMessage()：该方法的作用是返回描述异常的消息。
- String toString()：该方法的作用是返回关于异常的简短描述消息。
- void printStackTrace()：该方法的作用是显示异常及其栈轨迹。
- Throwable FillInStackTrace()：该方法的作用是重写异常的执行栈轨迹。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



26.5.8 config 对象

在有的应用中，需要获取某个 Servlet 或 JSP 页面的初始化参数，JSP 的 config 对象就能满足这种需求。config 对象是 javax.servlet.ServletConfig 类的实例，ServletConfig 封装了配置在 web.xml 中初始化 JSP 的参数。JSP 中通过 config 获取以下参数。

- getInitParameter（String name）：返回指定参数名字的初始参数值，这个参数是配置在 web.xml 中的。其语法为：public java.lang.String getInitParameter (java.lang.String name)。
- getInitParameterNames：返回该 JSP 中所有的初始参数名。它的语法为：public java.util.Enumeration getInitParameterNames ()。
- getServletName：返回所执行的 Servlet 的名字。它的语法为：public java.lang.String getServletName()
- getServletContext：返回所执行的 ServletContext 对象。

下面的实例就是使用 config 对象来获取某一个 Servlet 或者是 JSP 页面的配置信息。

【范例 26-10】示例代码 26-10 是一个 config 对象示例。

示例代码 26-10

```
<%@ page language="java" import="java.util.*" pageEncoding="utf-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    </head>
   <body>
   <div align="center">
         <h1>使用 config 获得服务信息</h1>
         当前 Servlet 名称：<%=config.getServletName()%>
         <table border="1" width="60%">
         <%
         //获取初始配置参数枚举
         Enumeration enumeration = config.getInitParameterNames();
         //判断获取初始配置参数枚举是否为空
         if(enumeration != null){
              //遍历获得的枚举
              while(enumeration.hasMoreElements()){
              //获取枚举中的参数名
              String enumeration_name = (String)enumeration.nextElement();
              //转换字符集，支持中文显示
              String name_zh = new String(enumeration_name.getBytes("utf-8"));

              //根据获取枚举中的参数名，获取枚举中的参数值
              String name_value = (String)config.getInitParameter(enumeration_
              name);
              //转换字符集，支持中文显示
              String value_zh = new String(name_value.getBytes("utf-8"));
         %>
              <tr>
                   <td align="center"><%=name_zh%></td>
                   <td align="center"><%=value_zh%></td>
              </tr>
         <%
              }
         }
         %>
</table>
</div>
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
36     </body>
37   </html>
```

【运行结果】程序运行结果如图 26-19 所示。



图 26-19　程序运行效果

【代码解析】在以上程序中，首先要初始化并获得参数枚举，设置参数以及对字符集的显示格式进行转换（第 11～24 行）。然后设置表格，并列出 name 和 value 值（第 27、28 行）。

26.6　习题

【习题 26-1】通过 JSP Scriptlets 对所声明的全局变量 a 和 b 进行运算，并将运算结果值赋予 c，使用 JSP 表达式在网页上输出变量 c 的值。

【分析】该题主要考查对 JSP 脚本的使用。

【核心代码】

```
<%
        int a = 2;              //声明变量a,其值为2
        int b = 1;              //声明变量b,其值为1
        int c = 0;              //声明变量c,其值为0
%>
    <%
        c = a + b;              //对a,b进行加法运算,并将值赋予c
    %>
    <%-- 输出c的值--%>
    <%= a + " + " + b + " = " + c + "<br>" %>

    <%
        c = a - b;              //对a,b进行减法运算,并将值赋予c
    %>
    <%= a + " - " + b + " = " + c + "<br>" %>

    <%
        c = a * b;              //对a,b进行乘法运算,并将值赋予c
    %>
    <%= a + " * " + b + " = " + c + "<br>" %>

    <%
        c = a / b;              //对a,b进行除法运算,并将值赋予c
    %>
    <%= a + " / " + b + " = " + c + "<br>" %>
```

【习题 26-2】编写一个 JSP 程序，通过使用 for 循环，在网页中输出一个 3 行 3 列的表格，

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

并设置表头信息。

【分析】该题主要考查对 JSP 页面的设置。

【核心代码】

```
<table border="1" width="60%">
    <tr>
        <th> 表头 1</th>
        <th> 表头 2</th>
        <th> 表头 3</th>
    </tr>
    <tr>
        <td> </td>
        <td> </td>
        <td> </td>
    </tr>
      <tr>
        <td> </td>
        <td> </td>
        <td> </td>
    </tr>
```

【习题 26-3】编写一个 JSP 程序，使用 page 指令的 import 属性导入“java.util.*”包，通过构造一个 LinkedList 实例对象并添加元素，最后在页面上显示 LinkedList 实例对象中的元素值。

【分析】该题主要考查对 page 指令的使用。

【核心代码】

```
<%@ page language="java" import="java.util.*"%>
```

【习题 26-4】编写一个 JSP 程序，使用实现跳转，并通过动作元素传递参数，在跳转页中接收参数并输出。

【分析】该题主要考查对 JSP 动作元素的使用，使用<jsp:forward>实现跳转，使用<jsp:plugin>元素进行参数的传递。

【核心代码】

```
<%@ page language="java" contentType="text/html; charset=utf-8" %>
<%@ page trimDirectiveWhitespaces="false"%>
<%
    out.clear();
    if("1".equals(request.getParameter("param"))){
%>
<jsp:forward page="/somepage.jsp">
    <jsp:param name="param1" value="value1"/>
    <jsp:param name="param2" value="value2"/>
</jsp:forward>
<%
    }
%>
<html>
<head>
<title>计数器</title>
<link rel='stylesheet' type='text/css' href='css/style.css'>
</head>

<body>
请使用地址栏参数 param=1 访问该页面。
</body>
```

【习题 26-5】编写一个 JSP 程序能够获得表单的所有参数，并获得参数的所有参数值。

【分析】该题主要考查对内置对象的使用。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

【核心代码】

```
<%@ page language="java" contentType="text/html;charset=gb2312" import="java.util.*"%>
<html>
    <head>
        <title>使用 request 对象获得所有的参数名称</title>
    </head>
    <body>
        <%-- 通过 request 对象的 getParameterNames --%>
        <%
            Enumeration e = request.getParameterNames();     // 获得所有的参数名称
            while(e.hasMoreElements()){                      //遍历 Enumeration
                String str = (String)e.nextElement();        //取出下一个元素值
                out.println(str);                            //打印输出变量名称
            }
        %>
    </body>
    </html>
```

【习题 26-6】编写一个 JSP 程序，使用<jsp:setProperty>为 Bean 的属性赋值，然后使用<jsp:getProperty>取出所有属性的值。

【分析】该题主要考查对内置对象的使用。

```
    <jsp:useBean id="book" class="org.gaozou.book.javaabc.exercises.ch26.Book" scope=
"page">
        <jsp:setProperty name="book" property="title" value="Java Professional"/>
        <jsp:setProperty name="book" property="author" value="George Li"/>
        <jsp:setProperty name="book" property="publish" value="China Machine Press"/>
        <jsp:setProperty name="book" property="price" value="40"/>
    </jsp:useBean>
    <h2>Book List</h2>
    <jsp:getProperty name="book" property="title"/>,
    <jsp:getProperty name="book" property="author"/>,
    <jsp:getProperty name="book" property="publish"/>,
    <jsp:getProperty name="book" property="price"/>
```

【习题 26-7】编写一个 JSP 程序，使用 response 设置相应的 MIME 类型。

【分析】该题主要考查对内置对象的使用。

```
  <%
     response.setCharacterEncoding("UTF-8");
     response.setContentType("text/html");
  %>
```

【习题 26-8】编写一个 JSP 程序，向 session 中保存用户信息，并取出信息；同时向 cookie 中写入信息，并取出。

【分析】该题主要考查对内置对象的使用。

```
    <jsp:useBean   id="user"   class="org.gaozou.book.javaabc.exercises.ch26.User"
scope="session">
        <jsp:setProperty name="user" property="name" value="Tom"/>
        <jsp:setProperty name="user" property="passwd" value="123456"/>
    </jsp:useBean>

    <h2>===========</h2>
    用户名: <jsp:getProperty name="user" property="name"/>,
    密码 : <jsp:getProperty name="user" property="passwd"/>
```

【习题 26-9】编写一个 JSP 程序，向 application 对象存取信息。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



【分析】该题主要考查对内置对象的使用。

```
    <jsp:useBean  id="about"  class="org.gaozou.book.javaabc.exercises.ch26.About"
scope="application">
        <jsp:setProperty name="about" property="name" value="Book Store"/>
        <jsp:setProperty name="about" property="copyright" value="Gaozou group"/>
    </jsp:useBean>

    <h2>============</h2>
    <jsp:getProperty name="about" property="name"/>,
    <jsp:getProperty name="about" property="copyright"/>
```

【习题 26-10】编写一个 JSP 程序，使用 config 对象获取 Web 应用的配置信息。

【分析】该题主要考查对内置对象的使用。

```
<%
    Enumeration e = config.getInitParameterNames();
    while(e.hasMoreElements()) {
        String paraName = (String)e.nextElement();
        out.println(paraName + "-----> " + config.getInitParameter(paraName)+"<br>");
    }
%>
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第 27 章　Servlet

Servlet 是在 JSP 之前推出的，但是因其使用不方便，所以 Sun 公司又推出了 JSP。在运行 JSP 前还需要编译成 Servlet，因此，非常有必要对 Servlet 的相关知识进行了解。本章主要通过示例展示 Servlet 的使用方法，并介绍其生命周期。



27.1　Servlet 简介

Servlet 是运行在服务器端的由 Web 服务器进行加载用来响应客户端请求的 Java 代码模块。Servlet 是一种独立于平台和协议的服务器端的 Java 应用程序，可以生成动态的 Web 页面。javax.servlet 和 javax.servlet.http 包为编写 Servlet 提供了接口和类。所有的 Servlet 都必须实现 Servlet 接口，该接口定义了生命周期方法。

Java Web 应用程序中所有的请求—响应都是由 Servlet 来完成的。Servlet 是 Java Web 的核心程序，所有的网址最终都交给 Servlet 来处理。Servlet 并没有 main 之类的执行方法。当用户访问服务器的时候，Tomcat 是通过调用 Servlet 的某些方法来完成整个处理过程的。

27.1.1　Servlet 接口

Servlet 实现了 javax.servlet.Servlet 接口，此接口中规定了特定的方法来处理特定的请求，开发中只需要实现 Servlet 的相关方法即可。在用户对 Web 程序进行访问时，Tomcat 来调用这些方法，以完成业务的处理。

Servlet 支持的 Web 访问方式包括：

- GET：此种方式常用来查询信息。根据 HTTP 协议，用 GET 方式查询时，Java 将提交的查询内容显示在浏览器地址栏中，并且 GET 方式提交的网址不能超过 256 个字符。
- POST：如果要提交大量的文本，需要使用 POST 方式。用 POST 方式提交数据时，数据信息不会在浏览器的地址栏中显示。用 POST 方式时，发送的命令要严格、复杂、安全，需要提供提交的数据类型及长度。数据类型有两种，一种是普通的文本数据（ASCII 码数据），类型为“application/x-www-form-urlencoded”；另一种是文件数据（二进制数据），类型为“multipart/form-data”。
- HEAD：查询文档头信息，服务器返回的是文件的类型、长度以及最后的修改时间。
- OPTIONS、PUT、DELETE、TRACE 属于被声明了，但是还没有被使用的 Web 访问方式。现在的浏览器中也很少使用这几种方式。

在以上七种访问方式中，最常见的是 GET 和 POST 方式，访问的形式为：void doXxx((HttpServletRequest request, HttpServletResponse response)，例如，doGET 或者 doPOST。其中有两个参数 HttpServletRequest 与 HttpServletResponse ，前者是客户端请求，后者是服务端请求。

27.1.2　目录结构

Web 程序一般都会部署在 Tomcat 的 webapps 下面，而在 Web 项目中，也有自己特定的目录机构：

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



- / web：应用根目录。
- /WEB-INF/（WEB-INF 文件夹）：Tomcat 会隐藏该文件夹下的所有文件及文件夹，保护它们不能通过浏览器直接访问。
- /WEB-INF/web.xml 文件：Web 程序最主要的配置文件。
- /WEB-INF/classes/：class 类文件都放在该文件下面，包括 Servlet 类。
- /WEB-INF/lib/：jar 文件都放在该文件下面。

27.1.3　编写 Servlet 程序

Servlet 是使用应用程序设计接口以及相关类和方法的 Java 程序，是位于 Web 服务器内部的服务器端的 Java 程序，Servlet 使用 Web 服务器加载，并且该 Web 服务器必须是包含支持 Servlet 的 Java 虚拟机。Servlet 技术的优点是高效性、方便、功能强大、可移植性好、节省投资。

不过编写 Servlet 却不容易，直接实现 Servlet 接口来编写 Servlet 很不方便，需要实现的方法太多。编写 Servlet 时直接继承 HttpServlet，并覆盖需要的方法即可。一般只覆盖 doGet 与 doPost 方法。

下面看一个示例，在前面创建的 MyJsp 项目中利用向导新建 Servlet，并且取类名为 FirstServlet。该 Servlet 实例覆盖了 doGet、doPost 和 getLastModified 三个方法。

【范例 27-1】示例代码 27-1 是一个 Servlet 示例。

示例代码 27-1

```
01  public class FirstServlet extends javax.servlet.http.HttpServlet implements
    javax.servlet.Servlet {
02  //Servlet 类的构造方法
03  public FirstServlet() {
04      super();
05      }
06
07      //覆盖父类的 doGet 方法
08  protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
09          this.log("执行 doGet 方法... "); // 调用 Servlet 自带的日志输出信息到控制台
10          this.execute(request, response); // 对 doGet 进行处理
11
12          // TODO Auto-generated method stub
13      }
14  //覆盖父类中的 doPost 方法
15      protected void doPost(HttpServletRequest request, HttpServletResponse
                              response) throws ServletException, IOException {

16      this.log("执行 doPost 方法... ");// 调用 Servlet 自带的日志输出信息到控制台
17              this.execute(request, response);    // 对 doPost 进行处理
18
19      }
20  //    执行方法
21  private void execute(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException{
22          response.setCharacterEncoding("UTF-8"); // 设置 response 的编码格式
23      request.setCharacterEncoding("UTF-8");       // 设置 request 的编码格式
24
25      String requestURI = request.getRequestURI();// 设置访问该 Servlet 的 URI
26
27      String method = request.getMethod();// 设置访问 Servlet 的方式，GET 或者 POST
28      String param = request.getParameter("param");//设置客户端提交的参数 param 值
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
29
30        response.setContentType("text/html");        // 设置文档类型为 HTML 类型
31        PrintWriter out = response.getWriter();
32    out.println("<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.01
                Transitional//EN\">");
33        out.println("<HTML>");
34        out.println("  <HEAD><TITLE>A Servlet</TITLE></HEAD>");
35        out.println("  <BODY>");
36        out.println("以 " + method + " 方式访问该页面。取到的 param 参数为：" + param +
37    "<br/>");
38
39        out.println("<form action='" + requestURI + "' method='get'><input
40                    type='text' name='param' value='param string'><input type=
      'submit' value='以 GET 方式查询页面 " + requestURI + "'></form>");
41            out.println("<form action='" + requestURI + "' method='post'><input
              type='text' name='param' value='param string'><input type='submit'
              value='以 POST 方式提交到页面 " + requestURI + "'></form>");
42            // 由客户端浏览器读取该文档的更新时间
43            out.println("<script>document.write('本页面最后更新时间：' +
44        document.lastModified); </script>");
45
46            out.println("  </BODY>");
47            out.println("</HTML>");
48            out.flush();
49            out.close();
50            }
51    }
```

【代码解析】在以上示例中，使用 HttpServlet 自带的日志函数来输出信息到控制台，并没有使用 System.out.println("")。执行 doGet 前会先执行 getLastModified，如果浏览器发现 getLastModified 返回的数值与上次访问时返回的数值相同，则认为该文档没有更新，浏览器采用缓存而不执行 doGet。如果 getLastModified 返回的值为-1，则认为是时刻更新的，总是执行该函数。

只有 Servlet 类的文件还不够，还需要配置 web.xml 文件，让浏览器访问这个 Servlet，也就是需要在 web.xml 文件中配置 Servlet 的类文件和访问方式，首先需要配置 Servlet 的名称和类名。

web.xml

```
01        <servlet>
02            <description>
03            </description>
04    <!--设置 name -->
05    <display-name>
06            FirstServlet</display-name>
07            <servlet-name>FirstServlet</servlet-name>
08    <!--设置路径 -->
09            <servlet-class>
10            cn.com.servlet.FirstServlet</servlet-class>
11        </servlet>
```

其中，在<servlet>和</servlet>之间配置的是 Servlet 信息，<servlet-name>项是配置 Servlet 的名称，<servlet-class>配置的是 Servlet 的类名。<servlet-name>必须保证在此配置文件中唯一。在 Servlet 中也还有其他的可选配置，如下所示：

```
01    <servlet>
02    <!—设置 Servlet 的 name 和路径 -->
03       <servlet-name>FirstServlet</servlet-name>
04       <servlet-class>com.helloweenvsfei.servlet.FirstServlet</servlet-class>
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
<init-param>
<!—设置 Servlet 的初始化参数 -->
    <param-name>message</param-name>
    <param-value>welcome to FirstServlet</param-value>
</init-param>
<!—设置 Servlet 的初始化参数 -->
   <init-param>
    <param-name>encoding</param-name>
    <param-value>utf-8</param-value>
   </init-param>
   <load-on-startup>1</load-on-startup>
  </servlet>
```

- <init-param>标签可以配置一个初始化参数。其中包括<param-name>和<param-value>参数。
- <load-on-startup>标签配置 Servlet 的加载方式，值可以为 0 或者 1。如果为 1，则在启动 Tomcat 的时候加载 Servlet；如果为 0，则在第一次请求该 Servlet 时加载 Servlet。

在配置好 Servlet 的名称和类名之后，还需要对 Servlet 的访问方式进行配置，访问方式的配置需要使用标签<servlet-mapping>，配置 FirstServlet 的代码如下：

```
<servlet-mapping>
    <servlet-name>FirstServlet</servlet-name>
    <url-pattern>/FirstServlet</url-pattern>
</servlet-mapping>
```

- <servlet-name>设置访问这种方式的 Servlet 名称，和前面<servlet>里配置的 Servlet 的名称相对应。
- <url-pattern>配置该 Servlet 的访问方式，在值前面加上 Web 应用程序的路径，然后加上服务器域名端口号信息，就是访问该 Servlet 的网址，在此标签中，允许使用“*”与“?”，前者表示任意长度的字符串，后者表示任意字符。

在<servlet-mapping>中，也可以配置多个<url-pattern>，例如，以下配置：

```
  <servlet-mapping>
<servlet-name>FirstServlet</servlet-name>
<!—设置多个 Servlet 访问 -->
    <url-pattern>/servlet/FirstServlet</url-pattern>
    <url-pattern>/servlet/FirstServlet.asp</url-pattern>
    <url-pattern>/servlet/FirstServlet.jsp</url-pattern>
    <url-pattern>/servlet/FirstServlet.php</url-pattern>
    <url-pattern>/servlet/FirstServlet.aspx</url-pattern>
  </servlet-mapping>
```

无论是使用以上哪一个路径访问，都会正常显示。

运行结果如图 27-1 所示。



图 27-1　Servlet 运行结果

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

27.2 Servlet 的生命周期

每一个 Servlet 都有一个生命周期，该生命周期由 Web 服务器进行维护，并且 Servlet 的生命周期遵循 Servlet 规范。服务器在启动或者第一次请求 Servlet 时初始化一个 Servlet 对象，然后使用这个 Servlet 去处理所有的客户端请求，在服务器关闭时才会去销毁这个 Servlet 对象。

每个 Servlet 首先要进行初始化，然后才能进行相应的逻辑操作。进行初始化时，需要调用的方法是 init（）方法。当加载 Servlet 时，首先由 Servlet 容器调用 Servlet 的 init（）方法进行初始化。在之前 JSP 对象的介绍中，有一个 JSP 内置对象 config，它主要用来配置 Servlet 的初始参数。

27.3 HttpServlet 类

27.3.1 doGet

当 FORM 中的 method 属性设置为“GET”时，浏览器就会以 GET 方式提交表单数据，表单会根据 Action 中的设置判断将数据提交到 Servlet 还是 JSP。

使用 GET 方式进行表单的提交时，浏览器会把表单的内容组织成一个字符串，变量之间用“&”进行连接，然后以 Servlet 路径加“？”和查询字符串的形式获取服务器内容。使用这种方式并不安全，因为所有需要传输的数据会显示在浏览器的地址栏上。



说明：如果不经过 FORM 表单，而是直接在浏览器地址栏中输入地址，也会被看做是使用 GET 的方式进行提交数据。

以下示例为演示 GET 提交表单数据，其中 JSP 文件是 get.jsp，Servlet 文件是 GetServlet.java。

【范例 27-2】示例代码 27-2 是一个 doGet 传值示例。

示例代码 27-2

```
<%@ page language="java" contentType="text/html; charset=GBK"
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=GBK">
<title>Insert title here</title>
</head>
<body>
<form action="/Forward/GetServlet" method="GET">
<div align="center"><br/>
    <fieldset style='width:80%'>
        <legend>填写用户信息</legend><br/>
        <div class='line'>
            <div align="left" class='leftDiv'>用户名：</div>
            <div align="left" class='rightDiv'>
                <input type="text" name="name" class="text" />
            </div>
        </div>

```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
22
23          <div class='line'>
24              <div align="left" class='leftDiv'>性别：</div>
25              <div align="left" class='rightDiv'>
26                  <input type="radio" name="sex" value="Male" id="sexMale">
27                  <label for="sexMale">男</label>
28                  <input type="radio" name="sex" value="Female" id="sexFemale">
29                  <label for="sexFemale">女</label>
30              </div>
31          </div>
32          <div class='line'>
33              <div align="left" class='leftDiv'>年龄：</div>
34              <div align="left" class='rightDiv'>
35                  <input type="text" name="age" class="text">
36              </div>
37          </div>
38                  <div class='line'>
39              <div align="left" class='leftDiv'>邮箱：</div>
40              <div align="left" class='rightDiv'>
41                  <input type="text" name="email" class="text" />
42              </div>
43          </div>
44
45          <div class='line'>
46              <div align="left" class='leftDiv'></div>
47              <div align="left" class='rightDiv'>
48              <br/><input type="submit" name=" btn" value="提交"
                class="button"><br/>
49              </div>
50          </div>
51      </fieldset>
52  </div>
53  </form>
54  </body>
55  </html>
```

web.xml

```
01      <servlet>
02          <description>
03          </description>
04  <!--设置 Display 属性 -->
05          <display-name>
06          GetServlet</display-name>
07  <!--设置 Servlet 名称 -->
08          <servlet-name>GetServlet</servlet-name>
09  <!--设置 Servlet 类地址 -->
10          <servlet-class>
11          cn.com.servlet.GetServlet</servlet-class>
12      </servlet>
13  <!--设置 Servlet 访问路径 -->
14      <servlet-mapping>
15          <servlet-name>GetServlet</servlet-name>
16          <url-pattern>/GetServlet</url-pattern>
17      </servlet-mapping>
```

GetServlet.java

```
01  public class GetServlet extends javax.servlet.http.HttpServlet implements
    javax.servlet.Servlet {
02      public GetServlet() {
03          super();
04      }
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
05      protected void doGet(HttpServletRequest request, HttpServletResponse
        response) throws ServletException, IOException {
06          response.setCharacterEncoding("GBK");   // 设置 response 编码
07          request.setCharacterEncoding("GBK");// 设置 request 编码
08          String name = request.getParameter("name"); // 获取用户姓名
09          String email = request.getParameter("email");// 获取用户密码
10          String sex = request.getParameter("sex");// 获取用户性别
11          int age = 0;
12          try{
13      age = Integer.parseInt(request.getParameter("age"));
        // 获取用户年龄，并转化为 int
14          }catch(Exception e){}
15  String[] interesting = request.getParameterValues("interesting");//取多个值
16          String area = request.getParameter("area"); // 取得 select 中的取值
17      // 取得文本框 textarea 中的值
18      String description = request.getParameter("description");
19          response.setContentType("text/html");       // 设置输出类型
20          PrintWriter out = response.getWriter();     // 获取 out 对象
21          out.println("<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.01
                    Transitional//EN\">");
22  out.println("<HTML>");
23  out.println("<HEAD><TITLE>感谢您提交信息</TITLE>");
24  out.println("<div align=\"center\"><br/>");
25  out.println("<fieldset style='width:90%'><legend>您的用户信息</legend><br/>");
26  out.println("<div class='line'>");
27  out.println("<div align='left' class='leftDiv'>您的姓名是：</div>");
28  out.println("<div align='left' class='rightDiv'>" + name + "</div>");
29  out.println("</div>");
30  out.println("<div class='line'>");
31  out.println("<div align='left' class='leftDiv'>您的性别是：</div>");
32  out.println("<div align='left' class='rightDiv'>" + sex + "</div>");
33  out.println("</div>");
34  out.println("<div class='line'>");
35  out.println("<div align='left' class='leftDiv'>您的年龄是：</div>");
36  out.println("<div align='left' class='rightDiv'>" + age + "</div>");
37  out.println("</div>");
38  out.println("<div class='line'>");
39  out.println("<div align='left' class='leftDiv'>您的邮箱是：</div>");
40  out.println("<div align='left' class='rightDiv'>" + email + "</div>");
41  out.println("</div>");
42  out.println("<div class='line'>");
43  out.println("<div class='line'>");
44  out.println("<div align='left' class='leftDiv'></div>");
45  out.println("<div align='left' class='rightDiv'>");
46  out.println("<br/><input type='button' name='btn' value='返回上一页'
47  onclick='history.go(-1); ' class='button'><br/>");
48  out.println("</div>");
49  out.println("</div>");
50  out.println("<BODY>");
51  out.println("</BODY>");
52  out.println("</HTML>");
53  out.flush();
54  out.close();
55  }
56 protected void doPost(HttpServletRequest request, HttpServletResponse response)
   throws ServletException, IOException {
57      response.setCharacterEncoding("GBK");// 设置输出信息的编码格式
58      response.getWriter().println("请使用 GET 方式提交数据。");// 设置输出提示信息
59  }
60  }
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

【运行结果】运行结果如图 27-2 所示。

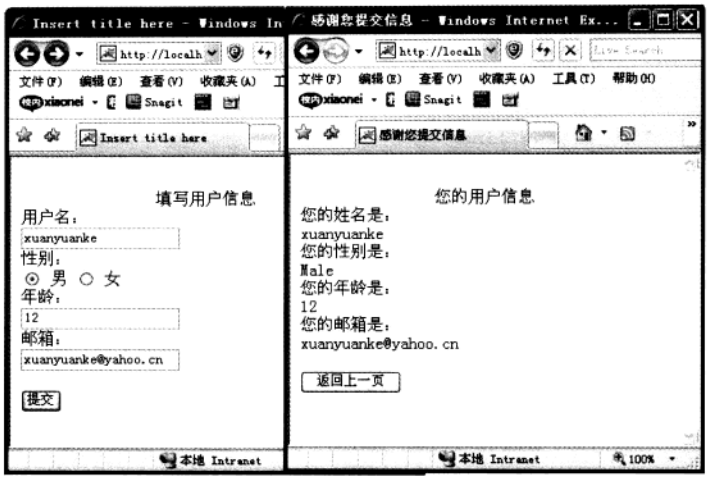


图 27-2　使用 GET 方法进行表单的提交效果图

27.3.2　doPost

使用 POST 提交表单时，只需要将 FORM 中的 methods 属性设置为 POST，浏览器就会以 POST 的方式提交表单内容。在 POST 提交表单时，表单的内容不会在浏览器中显示，因此，使用 POST 提交方式比较安全，适用于提交密码等信息。

Servlet 可以通过 HttpServletRequest 对象的 getPrarameter(String param)方法来获取 param 对应的参数值。下面的示例是通过 POST 的方式来提交用户信息的表单，其中有用于填写信息的页面“post.html”和用于进行数据处理的 Servlet“PostServlet.java”。

【范例 27-3】示例代码 27-3 是一个 doPost 传值示例。

示例代码 27-3

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>提交用户信息</title>
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
<style><!-- css 略 --></style>
</head>
<body>
<form action="/Forward/PostServlet" method="POST">
<div align="center"><br/>
    <%--  --%>
    <fieldset style='width:80%'>
            <%-- 输出提示信息 --%>
        <legend>填写用户信息</legend><br/>
        <div class='line'>
            <%-- 设置用户名行  --%>
            <div align="left" class='leftDiv'>用户名：</div>
            <div align="left" class='rightDiv'>
                <input type="text" name="name" class="text" />
            </div>
        </div>
            <%-- 设置密码行 --%>
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
23          <div class='line'>
24              <div align="left" class='leftDiv'>密码：</div>
25              <div align="left" class='rightDiv'>
26                  <input type="password" name="password" class="text" />
27              </div>
28          </div>
29              <%-- 设置密码验证行 --%>
30          <div class='line'>
31              <div align="left" class='leftDiv'>请再次输入密码：</div>
32              <div align="left" class='rightDiv'>
33                  <input type="password" name="passwordConfirm" class="text" />
34              </div>
35          </div>
36              <%-- 设置性别选择行 --%>
37          <div class='line'>
38              <div align="left" class='leftDiv'>性别：</div>
39              <div align="left" class='rightDiv'>
40                  <input type="radio" name="sex" value="男" id="sexMale">
41                  <label for="sexMale">男</label>
42                  <input type="radio" name="sex" value="女" id="sexFemale">
43                  <label for="sexFemale">女</label>
44              </div>
45          </div>
46              <%-- 设置年龄输入行 --%>
47          <div class='line'>
48              <div align="left" class='leftDiv'>年龄：</div>
49              <div align="left" class='rightDiv'>
50                  <input type="text" name="age" class="text">
51              </div>
52          </div>
53              <%-- 设置专业输入行 --%>
54          <div class='line'>
55              <div align="left" class='leftDiv'>专业</div>
56              <div align="left" class='rightDiv'>
57                  <input type="checkbox" name="interesting" value="Java"
58  id="i1">
59                  <label for="i1">Java</label>
60                  <input type="checkbox" name="interesting" value="C#" id="i2">
61                  <label for="i2">C#</label>
62                  <input type="checkbox" name="interesting" value="C++"
63  id="i3">
64                  <label for="i3">C++</label>
65              </div>
66          </div>
67              <%-- 设置地址选择 --%>
68          <div class='line'>
69              <div align="left" class='leftDiv'>住址：</div>
70              <div align="left" class='rightDiv'>
71                  <select name="area">
72              <%-- 加入省份选择 --%>
73                      <option>---省份---</option>
74                      <optgroup label="北京市">
75              <%-- 加入地区选择 --%>
76                          <option value="北京市海淀区">海淀区</option>
77                          <option value="北京市朝阳区">朝阳区</option>
78                          <option value="北京市东城区">东城区</option>
79                          <option value="北京市西城区">西城区</option>
80                      </optgroup>
81              <%-- 加入省份选择 --%>
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
                        <optgroup label="山东省">
                <%-- 加入地区选择 --%>
                            <option value="山东省济南市">济南市</option>
                            <option value="山东省德州市">德州市</option>
                            <option value="山东省泰安市">泰安市</option>
                            <option value="山东省潍坊市">潍坊市</option>
                            <option value="山东省淄博市">淄博市</option>
                            <option value="山东省烟台市">烟台市</option>
                            <option value="山东省济宁市">济宁市</option>
                        </optgroup>
                    </select>
                </div>
            </div>
            <div class='line'>
                <div align="left" class='leftDiv'>自我描述：</div>
                <div align="left" class='rightDiv'>
                    <textarea name="description" rows="8"> </textarea>
                </div>
            </div>
                <%-- 设置提交按钮 --%>
            <div class='line'>
                <div align="left" class='leftDiv'></div>
                <div align="left" class='rightDiv'>
                    <br/><input type="submit" name=" btn" value="提交"
class="button"><br/>
                </div>
            </div>
        </fieldset>
</div>
</form>
</body>
</html>
```

web.xml

```
<servlet>
            <description>
            </description>
<!—设置 Display 属性 -->
            <display-name>
            PostServlet</display-name>
<!—设置 Servlet 名称 -->
            <servlet-name>PostServlet</servlet-name>
<!—设置 Servlet 类地址 -->
            <servlet-class>
            cn.com.servlet.PostServlet</servlet-class>
</servlet>
<!—设置 Servlet 访问路径 -->
<servlet-mapping>
            <servlet-name>PostServlet</servlet-name>
            <url-pattern>/PostServlet</url-pattern>
</servlet-mapping>
```

PostServlet.java

```
public class PostServlet extends HttpServlet
    {
    public void doGet(HttpServletRequest request, HttpServletResponse
    response)
                throws ServletException, IOException    // 使用 GET 执行该方法
    {
        response.setCharacterEncoding("UTF-8");     // 设置输出信息的编码格式
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
07          response.getWriter().println("请使用 POST 方式提交数据。");
08                                                  // 设置输出提示信息
09      }
10      public void doPost(HttpServletRequest request, HttpServletResponse
11  response)
12              throws ServletException, IOException
13       {
14          response.setCharacterEncoding("UTF-8");      // 设置 response 编码
15          request.setCharacterEncoding("UTF-8");       // 设置 request 编码
16
17          String name = request.getParameter("name"); // 获取用户姓名
18          String password = request.getParameter("password");// 获取用户密码
19          String sex = request.getParameter("sex");   // 获取用户性别
20
21          int age = 0;
22          try{
23          // 获取用户年龄，并转化为 int
24              age = Integer.parseInt(request.getParameter("age"));
25          }catch(Exception e){}
26          //取多个值
27          String[] interesting = request.getParameterValues("interesting");
28          String area = request.getParameter("area"); // 取得 select 中的取值
29          // 取得文本框 textarea 中的值
30          String description = request.getParameter("description");
31
32          response.setContentType("text/html");   // 设置输出类型
33      PrintWriter out = response.getWriter();     // 获取 out 对象
        out.println("<!DOCTYPE HTML PUBLIC \"-      //W3C//DTD HTML 4.01
                    Transitional//EN\">");
34          out.println("<HTML>");
35          out.println("<HEAD><TITLE>感谢您提交信息</TITLE>");
36          out.println("<div align=\"center\"><br/>");
37          out.println("<fieldset style='width:90%'><legend>您的用户信息
                      </legend><br/>");
38          out.println("<div class='line'>");
39          out.println("<div align='left' class='leftDiv'>您的姓名是：</div>");
40          out.println("<div align='left' class='rightDiv'>" + name + "</div>");
41          out.println("</div>");
42          out.println("<div class='line'>");
43          out.println("<div align='left' class='leftDiv'>您的密码是：</div>");
44          out.println("<div align='left' class='rightDiv'>" + password + 45
                        "</div>");
46          out.println("</div>");
47          out.println("<div class='line'>");
48          out.println("<div align='left' class='leftDiv'>您的性别是：</div>");
49          out.println("<div align='left' class='rightDiv'>" + sex + "</div>");
50          out.println("</div>");
51          out.println("<div class='line'>");
52          out.println("<div align='left' class='leftDiv'>您的年龄是：</div>");
53          out.println("<div align='left' class='rightDiv'>" + age + "</div>");
54          out.println("</div>");
55          out.println("<div class='line'>");
56          out.println("<div align='left' class='leftDiv'>您的特长是：</div>");
57          out.println("<div align='left' class='rightDiv'>");
58          for(String str : interesting){ out.println(str + ", "); }
59          out.println("</div>");
60          out.println("</div>");
61          out.println("<div class='line'>");
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
62            out.println("<divalign='left'class='leftDiv'>你的自我描述：</div>");
63        out.println("<div align='left' class='rightDiv'>" + description +
                  "</div>");
64            out.println("</div>");
65            out.println("<div class='line'>");
66            out.println("<div align='left' class='leftDiv'></div>");
67            out.println("<div align='left' class='rightDiv'>");
68            out.println("<br/><input type='button' name='btn' value='返回上一页'
                      onclick='history.go(-1); ' class='button'><br/>");
69            out.println("</div>");
70            out.println("</div>");
71            out.println("<BODY>");
72            out.println("</BODY>");
73            out.println("</HTML>");
74            out.flush();
75            out.close();
76        }
77    }
```

【运行结果】运行效果如图 27-3 所示。

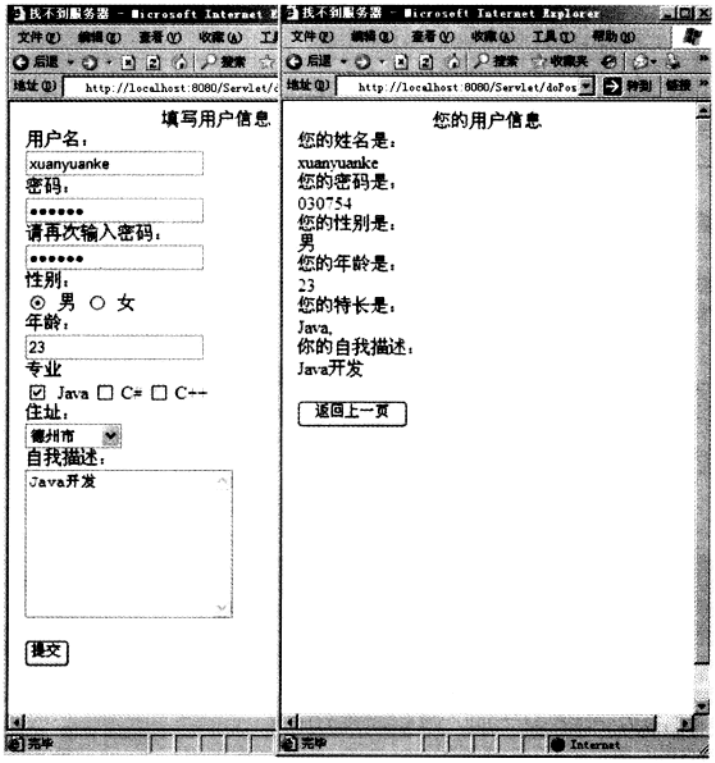


图 27-3　doPost 示例效果

【代码解析】如果要提交大量的文本，需要使用 POST 方式。用 POST 提交数据时，数据信息不会在浏览器的地址栏中显示。用 POST 方式发送的命令要严格、复杂、安全，需要提供提交的数据类型及长度。数据类型有两种，一种是普通的文本数据（ASCII 码数据），类型为“application/x-www-form-urlencoded”。另一种是文件数据（二进制数据），类型为“multipart/form-data”。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

27.4 HttpServletResponse 类及状态响应码

所有的 Servlet 响应都要实现 ServletResponse 接口，在 ServletResponse 接口中有一些方法，这些方法如表 27-1 所示。

表 27-1 HttpServletResponse 方法

方 法 名	说 明
getWriter	获取 PrintWriter 对象，用来输出字符数据，并且可以从 Servlet 中通过 getOutputStream 方法来获取 ServletoutputStream 对象。使用 ServletOutputStream 对象可以对字符数据进行输出，也可以对 MIME 格式的二进制数据进行输出
setContentType	此方法可以在响应中表明内容的格式长短
setBufferSize	此方法可以对 Web 容器的缓冲器大小进行设置
getBufferSize	此方法可以返回 Web 容器的缓冲器大小
resetBuffer	此方法可以对缓冲器进行清空并重置
Reset	此方法可以对缓冲器和状态的信息进行清空
flushBuffer	此方法可以将缓冲器内的所有输出内容传递给客户
isCommitted	此方法可以对响应是否已经被履行进行判断
setLocal	此方法可以对响应的地域信息进行设定，主要在 Web 程序的国际化时使用



注意：setLocal 方法应该使用在 getWriter 之前，是默认的 Web 服务器的地域信息。

HttpServletResponse 继承了 ServletResponse 接口，并提供了与 HTTP 协议有关的方法，这些方法的主要功能是设置 HTTP 状态码和管理 Cookie，如表 27-2 所示。

表 27-2 HTTP 协议方法

方 法 名	说 明
setStatus	此方法可以设置 Servlet 向客户端返回的状态码，设置没有出错的状态
sendError	此方法可以设置 Servlet 运行出错时的状态码，可以有两个不同的参数，如 sendError(int sc)和 sendError(int sc,String msg)，其中前者设置错误状态代码，后者除了设置状态代码外，还向客户发出错误信息
setHeader	此方法可以指定特定名称的头信息的值，例如 addDateheader、addIntHeader、setDateHeader、setIntHeader 可以加入或指定特定时间或整数型头信息
addCookie	此方法可以在 Web 服务器响应中加入 Cookie 对象，并且在浏览器中保存这个对象，Cookie 机制也会用来维护会话状态
sendRedirect	此方法用于对客户发出的临时重新导向进行响应

在浏览网页信息时，我们经常会看到如 500 错误之类的信息，这些错误代码所代表的含义是一个 Web 开发人员所必须了解的。其中 HTTP 的响应代码是三位数字值，第一个数字定义了结果代码的类别，这些错误代码被分为以下五类。

- 100～199：信息——请求被接收，正在处理。
- 200～299：成功——动作被成功接收、理解并接受。
- 300～399：重定向——必须采取其他动作才能完成请求。
- 400～499：客户机错误——请求包含不良语法或无法完成。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



- 500～599：服务器错误——服务器无法处理明显有效的请求。

最常见的几种错误信息代号及含义如表 27-3 所示。

表 27-3　错误代码表

错误代号	说　明
200	表示请求成功
302	表示临时移动
400	表示语法错误
401	表示未经授权
403	表示禁止访问
404	表示未发现请求 URL
500	表示内部服务器错误



27.5　Servlet 中的请求与响应

当客户端向服务器发送请求时，服务器端会根据请求进行处理，然后做出响应，再返回给客户端，这就完成了一次 Web 操作过程。这个过程就是通过请求来分析客户需要的信息并做出相应的操作，然后通过一系列的处理，最后将响应结果显示给客户。

服务器对客户端浏览器做出的响应被封装成为一个 HttpServletResponse 对象。要对浏览器进行操作，只需要操作 HttpServletResponse 对象就可以了。可以通过 HttpServletResponse.getWriter()方法来获得一个 PrintWriter 对象，然后使用该对象输出信息。以下示例为生成随机图片的验证码，其中 Servlet 为 PrintServlet.java，显示页面是 Identity.html。

【范例 27-4】示例代码 27-4 是一个请求与响应示例。

示例代码 27-4

```
01  public class PrintServlet extends javax.servlet.http.HttpServlet
02  {
03          //设置要随机产生的数
04          public static final char[] CHARS = { '2', '3', '4', '5', '6', '7', '8',
05                  '9', 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'J', 'K', 'L',
06                  'M', 'N', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y',
                    'Z' };
07                  public static Random random = new Random(); //随机产生数
08          public static String getRandomString()
09          {    // 随机获取 6 位数
10              StringBuffer buffer = new StringBuffer();   // 字符串缓存
11              for (int i = 0; i < 6; i++) {               // 依次循环六次
12          // 每次获取一个随机产生的字符
13          buffer.append(CHARS[random.nextInt(CHARS.length)]);
14              }
15              return buffer.toString();
16          }
17          public static Color getRandomColor()
18          {    // 获取随机产生的颜色
19          return new Color(random.nextInt(255), random.nextInt(255),
                            random.nextInt(255));
20      }
21      public static Color getReverseColor(Color c)
22      {       // 返回颜色的反色
23          return new Color(255 - c.getRed(), 255 - c.getGreen(), 255 - c.getBlue());
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
    }
    public void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("image/jpeg");          // 设置输出类型
        String randomString = getRandomString();        // 随机字符串
        request.getSession(true).setAttribute("randomString", randomString);
            // 放到 session 中
        int width = 200;      // 设置图片宽度
        int height = 50;      // 设置图片高度
        Color color = getRandomColor();// 设置随机产生的颜色，用于设定背景色
        Color reverse = getReverseColor(color);     // 反色，用于前景色
    BufferedImage bi = new BufferedImage(width, height, // 创建一个彩色的图片
                BufferedImage.TYPE_INT_RGB);
        Graphics2D g = bi.createGraphics();         // 获取绘图对象
        g.setFont(new Font("宋体", Font.BOLD, 16)); // 设置随机数的字体
        g.setColor(color); // 设置随机数的颜色
        g.fillRect(0, 0, width, height); // 绘制随机数的背景
        g.setColor(reverse); // 设置随机数的颜色
        g.drawString(randomString, 18, 20);     // 绘制随机字符
        for (int i = 0, n = random.nextInt(100); i < n; i++) {
            // 设置随机数中最都有 100 个噪声点
g.drawRect(random.nextInt(width), random.nextInt(height), 1, 1);
            // 随机噪声点
            }
            // 将随机数转成 JPEG 图片格式
            ServletOutputStream out = response.getOutputStream();
            // 设置编码器
            JPEGImageEncoder encoder = JPEGCodec.createJPEGEncoder(out);
            encoder.encode(bi);   // 对图片进行编码
            out.flush();          // 输出到客户端
        }
```

web.xml

```
<servlet>
        <description>
        </description>
<!—设置 Servlet 的 Display 属性 -->
        <display-name>
        PrintServlet</display-name>
        <servlet-name>PrintServlet</servlet-name>
<!—设置 Servlet 的地址 -->
        <servlet-class>
        cn.com.jsp.PrintServlet</servlet-class>
</servlet>
<!—设置 Servlet 的访问路径 -->
    <servlet-mapping>
        <servlet-name>PrintServlet</servlet-name>
        <url-pattern>/PrintServlet</url-pattern>
    </servlet-mapping>
```

Identity.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
09  <script>
10  <%-- 使用 script 设置方法 --%>
11  function reloadImage() {
12      document.getElementById('btn').disabled = true;
13      document.getElementById('identity').src='servlet/IdentityServlet?ts=' +
14  new Date().getTime();
15  }
16  </script>
17  <%-- 调用方法 --%>
18  <img src="PrintServlet" id="PrintServlet" onload="btn.disabled = false; " />
19  <input type=button value=" 换个图片 " onclick="reloadImage()" id="btn">
20
21  </body>
22  </html>
```

【运行结果】运行效果如图 27-4 所示。



图 27-4　随机图片效果

27.6　Servlet 之间的跳转

Servlet 和 Servlet 之间可以实现跳转，Servlet 的跳转可以将一个项目的模块进行划分，例如，在一个项目开发中，使用业务时由一个 Servlet 处理，使用数据库的操作时可以用另外的 Servlet 进行处理，然后跳转到另外一个 Servlet 中将处理结果打印出来。

27.6.1　Forward

Forward 是通过 RequestDispatcher 对象的方法 forward(HttpServletRequest req, HttpServlet Response res)来实现的。RequestDispatcher 可以通过 HttpServletRequest 的 getRequestDispatcher() 方法获得。使用方法如下：

```
    RequestDispatcher dispatcher = request.getRequestDispatcher("/servlet/First Serv-
let");
    dispatcher.forward(request, response);
```

其中的路径以“/”开始，“/”表示项目的根目录。其中的路径是在 XML 中配置的<url-pattern>中的路径。如果在 XML 中配置为：

```
01      <servlet>
02          <description>
03          </description>
04          <display-name>
05          Servlet_init</display-name>
06  <!--设置 Servlet 名 -->
07          <servlet-name>Servlet_init</servlet-name>
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
<!--设置 Servlet 类地址 -->
        <servlet-class>
        cn.com.servlet.Servlet_init</servlet-class>
    </servlet>
<!--设置 Servlet 访问路径 -->
    <servlet-mapping>
        <servlet-name>Servlet_init</servlet-name>
        <url-pattern>/Servlet/Servlet_init</url-pattern>
        </servlet-mapping>
```

则跳转方法应为：

```
RequestDispatcher dispatcher = request.getRequestDispatcher("/Servlet/Servlet_init
                               ");
dispatcher.forward(request, response);
```

Forward 跳转页是 MVC 框架中比较常用的一种技术，使用 Forward 可以进行 Servlet 之间的跳转，也可以进行 Servlet 和 JSP 页面之间的跳转，还可以跳转到一个文件。其中最常见的是 Servlet 和 JSP 之间的跳转。在 MVC 框架中，通常使用一个 Servlet 来集中处理请求，然后跳转到被响应的 Servlet，或者只在 Servlet 中进行业务的处理，然后将结果返回到 JSP 中。

下面的示例就是从 Servlet 跳转到 JSP 页面中，从而获取本地的时间信息，并打印输出。其中分为 ForwardServlet.jsva 和 forward.jsp。

【范例 27-5】示例代码 27-5 是一个 Forward 跳转示例。

示例代码 27-5

```
public class ForwardServlet extends HttpServlet
{
    public void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException
    {
        String destination = request.getParameter("destination");
        //判断是否 file=destination
        if("file".equals(destination)){ // 跳转到 /WEB-INF/web.xml
            RequestDispatcher d =
    request.getRequestDispatcher("/WEB-INF/web.xml");
            d.forward(request, response);
        }
        else if("jsp".equals(destination)){ // 跳转到 /forward.jsp
    //向 JSP 页面传递一个 Date 对象
            request.setAttribute("date", new Date());
            RequestDispatcher dispatcher =
    request.getRequestDispatcher("/forward.jsp");
            dispatcher.forward(request, response);
        }
        //否则执行以下操作
        else{
            response.setCharacterEncoding("GBK");
    response.getWriter().println("缺少参数。用法：" + request.getRequestURL() +
                                 "?destination=jsp 或者 file 或者 servlet ");
        }
    }
}
```

Forward.jsp

```
<%@ page language="java" contentType="text/html; charset=GBK" %>
<jsp:directive.page import="java.util.Date"/>
<jsp:directive.page import="java.text.SimpleDateFormat"/>
<%
    Date date = (Date)request.getAttribute("date");
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
06  %>
07  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
08  <HTML>
09  <HEAD>
10  <TITLE>Forward 跳转</TITLE>
11  <link rel='stylesheet' type='text/css' href='<%= request.getContextPath()
12  %>/css/style.css'>
13  </HEAD>
14  <BODY>
15  <%-- 设置跳转路径 --%>
16  <div align='center'><br/><legend>Forward 跳转</legend><br/>
17  <div style='line'>
18      <div>从 ForwardServlet 中取到的 Date 为</div>
19      <div><%= new SimpleDateFormat("yyyy-MM-dd HH:mm:ss:SSS").format(date)
20  %></div>
21  </div><br/>
22
23  </div>
24  </BODY>
25</HTML>
```

【运行结果】以上示例的运行结果如图 27-5 所示。



图 27-5 ForwardServlet 跳转效果图

从以上示例中可以看到，在使用 Forward 跳转 Servlet 时，地址栏显示的是跳转前的 Servlet 地址，而不是跳转后的地址，因为这个跳转过程是在服务器端实现的，并不是在客户端实现的，所以客户端并不知道这个跳转动作。

注意：如果使用 Forward 方式进行跳转，在跳转前最好不要使用任何输出，例如，使用 ort.println()语句等，否则会抛出 IllegalStateException。

27.6.2 Redirect

重定向是利用服务器返回的状态码来实现的。客户端浏览器请求服务器的时候，服务器端会返回一个状态码。服务器端通过 HttpServletResponse 的 setStatus(int statuss)方法设置状态码。如果服务器返回 301 或者 302，则浏览器会到新的网址重新请求该资源，其中分为五种状态码。

- 1xx：信息状态码。表示该请求已经被接受，正在被处理。
- 2xx：正确状态码。表示该请求已经被正确接受并处理，没有错误等发生。例如，200 表示一切正确。
- 3xx：重定向状态码。例如 301、302 表示该资源已经不存在或者换了地址，客户端需要重新定向到一个新的资源。服务器响应中会附带这个新资源的地址。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

- 4xx：请求错误。例如 401 表示没有权限访问，404 表示资源不存在，405 表示访问方式错误（如 Servlet 只接受 GET、POST 方式访问，但是客户端却以 DELETE 方式访问）。
- 5xx：服务器错误。例如 500 表示程序出现异常而中途停止运行。

以下代码就是利用资源重定向：

```
response.setStatus(HttpServletResponse.SC_MOVED_TEMPORARILY);  // 设置状态码为 302
response.setHeader("Location", "http://www.helloweenvsfei.com"); // 新网址
```

其中代码 SC_MOVED_TEMPORARILY 表示的是 302，因为 HttpServletResponse 类把常用的状态码封装成了静态常量。其中 sendRedirect 方法中封装了 setStatus 和 setHeader 方法。

下面的例子是利用一个 Servlet 来统计下载文件的下载次数。要下载的文件以及下载次数都保持在一个 Map 中。当单击下载的时候，doGet 获取到文件名、更新下载的次数，并重定向到要下载的文件，达到统计下载次数的目的。

【范例 27-6】示例代码 27-6 是一个资源重定向示例。

示例代码 27-6

```
01  public class Redirect extends HttpServlet
02  {
03
04      Map<String, Integer> map = new HashMap<String, Integer>();
05
06      @Override
07      public void init() throws ServletException {
08          map.put("/download/1.txt", 0);
09          map.put("/download/2.txt", 0);
10          map.put("/download/3.txt", 0);
11      }
12
13      public void doGet(HttpServletRequest request, HttpServletResponse response)
14              throws ServletException, IOException
15      {
16          String filename = request.getParameter("filename");
17
18          if(filename != null){ // 设置统计下载次数
19              int hit = map.get(filename);   // 取得已经下载次数
20              map.put(filename, ++ hit);     // 已经下载的次数+1，并保存
21              // 重定向到文件
22              response.sendRedirect(request.getContextPath()   +   filename);
23
24          }
25          else{          // 显示下载的次数
26              //以下为 JSP 输出
27              response.setCharacterEncoding("UTF-8");
28              response.setContentType("text/html");
29
30              PrintWriter out = response.getWriter();
            out.println("<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.01
                    Transitional//EN\">");
31              out.println("<HTML>");
32              out.println("  <HEAD><TITLE>文件下载</TITLE></HEAD>");
33              out.println("<link rel='stylesheet' type='text/css'
                        href='../css/style.css'>");
34              out.println("  <BODY><br/>");
35              out.println("<legend>文件下载</legend>");
36              out.println("<table width=100%>");
37              out.println("<tr>");
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
38                  out.println("     <td><b>文件名</b></td>");
39                  out.println("     <td><b>下载次数</b></td>");
40                  out.println("     <td><b>下载</b></td>");
41                  out.println("</tr>");
42
43                  for(Entry <String, Integer> entry : map.entrySet()){
44                      out.println("<tr>");
45                      out.println("<td>" + entry.getKey() + "</td>");
46                  out.println("<td>" + entry.getValue() + "</td>");
47                  out.println("<td><a href='" + request.getRequestURI() +
                                "?filename=" + entry.getKey() + "' target='_blank'
                                onclick='location=location; '>下载</a></td>");
48                      out.println("</tr>");
49                  }
50
51                  out.println("</table>");
52                  out.println("</legend>");
53                  out.println("  </BODY>");
54                  out.println("</HTML>");
55                  out.flush();
56                  out.close();
57              }
58          }
59          @Override
60          public void destroy() {
61              map = null;
62          }
63      }
```

【运行结果】运行结果如图 27-6 所示。

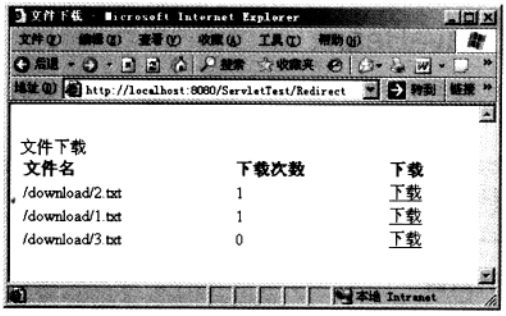


图 27-6　资源重定向效果图

27.6.3　Refresh

使用自动刷新功能可以实现自动跳转到另一个页面，也可以实现相隔一段时间之后刷新本页面，在 Servlet 中可通过 HttpServletResponse 来设置 Header 属性实现自动刷新效果。方法代码为：

```
response.setHeader("Refresh", "1000; URL=http://localhost:8080/servlet/example.
htm");
```

其中，1000 为设置刷新的时间，单位是毫秒。URL 参数指定的网址就是 1 秒钟之后跳转的页面。当 URL 设置的路径为 Servlet 自己的路径时，就会每隔一秒钟自动刷新本页面一次。这种情况可以用于设置邮箱，以达到及时收邮件的效果。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

27.7 JavaBean

JavaBean 可以很好地将业务逻辑与显示代码相分离，使得业务逻辑的工程师与界面工程师各司其职，减少协同开发的成本。本节主要介绍 JavaBean 的使用。

27.7.1 使用 JavaBean

JavaBean 类就是普通的 Java 类，也被称为 POJO（普通 Java 对象，Plain Ordinary Java Object），是 Java 程序设计中广泛应用的一种设计模式。JavaBean 中只有私有属性和与之对应的 setter/getter 方法。

JavaBean 是描述 Java 的软件组件模型，类似于 Microsoft 的 COM 组件概念，在 Java 模型中，通过 JavaBean 可以扩充 Java 程序的功能；通过 JavaBeans 的组合可以快速生成新的应用程序。对于程序员来说，使用 JavaBean 可以实现代码的重复利用，使得程序的维护更加简单。

JavaBean 组件是根据一系列规定设计出来的普通 Java 类，这些规定就是 JavaBean 规范，以下是 JavaBean 中的编码规则。

- JavaBean 类必须是一个公共类，类都必须设置为 public。
- 一个 JavaBean 必须有一个空的构造函数，必须有一个不带有任何参数的功用构造器，这个构造器应该通过调用其他的特性设置方法来设置特性的默认值。
- 一个 JavaBean 中不应该有公共的实例变量，类变量都应该为 private。
- 一个 JavaBean 中持有值应该通过一组存取方法来访问，应该都带有相匹配的 getter 和 setter 方法的专用示例变量。

27.7.2 JSP 中的 JavaBean

JavaBean 所实现的代码重用使得其在 JSP 技术中得到了很好的应用，下面介绍 JSP 中 JavaBean 的使用。

在 JSP 中使用 JavaBean 时，需要在 JSP 页面中对 JavaBean 类进行声明，声明 Javabean 对象时需要使用 scope 属性指定其有效范围，JavaBean 在 JSP 页面中有四种属性选择，分别是 page、request、session 和 application。

1. 利用 scope 实现计数器

在前面的例子中，scope 的属性是“page”，如果将 scope 设置为 page，则只有 JSP 脚本才可以使用 JavaBean 对象。scope 的属性还可以设置为 request、session、application，它们的范围是依次增大的。

设置为 request 时，只能在 request 内使用该 Bean 对象；设置为 session 时，只能在 session 内使用该 Bean 对象；设置为 application 时，在 Web 程序内均可以使用。

可通过设置 JavaBean 不同的 scope 属性来实现特殊的效果，例如，制作计数器功能时，需要将 scope 设置为 session，才能够统计出特定用户的访问次数，如果将属性设置为 application，就可以统计出所有的用户访问次数，计数器分为 Counter.java 和 counter.jsp。

【范例 27-7】示例代码 27-7 是一个计数器示例。

示例代码 27-7

```
01  public class Counter
02  {
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
    private int count;                          // 设置计数器
    public int getCount()
    {
        return ++count;                         // 用户每访问一次，计数器自加 1
    }
    public void setCount(int count) {           //计数器的 set 方法
        this.count = count;
    }
}
```

counter.jsp

```
<%@ page language="java" contentType="text/html; charset=utf-8" %>
<html>
<head>
<title>计数器</title>
<link rel='stylesheet' type='text/css' href='css/style.css'>
</head>
<body><br/>

<%-- 定义一个 session 范围内的计数器来记录个人的访问信息 --%>
<jsp:useBean id="personCount" class="cn.com.jsp.Counter" scope="session" />

<%-- 定义一个 application 范围内的计数器，用来记录所有的访问信息 --%>
<jsp:useBean id="totalCount" class="cn.com.jsp.Counter" scope="application" />

<div align="center">
    <form action="method.jsp" method="get">
    <legend>计数器</legend>
    <table align="center" width="400">
    <tr>
    <td width=150 align="right" style="font-weight:bold; ">您的访问次数：</td>
            <td>
            <%--用于获得个人的访问次数 --%>
        <jsp:getProperty name="personCount" property="count" /> 次
            </td>
        </tr>
        <tr>
    <tdwidth=150 align="right" style="font-weight:bold; ">总共的访问次数：</td>
            <td>
            <%--用于获得所有人的访问次数 --%>
            <jsp:getProperty name="totalCount" property="count" /> 次
        </td>
        </tr>
    </table>

    </form>
</div>
</body>
</html>
```

【运行结果】运行结果如图 27-7 所示。

2. 设置页面有效期 Page

Page 是 JavaBean 中有效范围最小的，值代表目前访问的页面。下面通过一个示例进行演示说明，这个示例也是一个页面访问次数的统计，分为一个 JSP 页面“Page.jsp”和一个 JavaBean 类“CountNum.java”，其中前者负责对 JavaBean 的调用，后者负责逻辑运算和处理。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



图 27-7　计数器运行效果图

【范例 27-8】示例代码 27-8 是一个页面有效期示例。

示例代码 27-8

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<%-- 设置颜色 --%>
<body bgcolor="#FFFFFF">
<%-- 设置 JavaBean 的地址及 id--%>
<jsp:useBean id="visitcount" class="cn.com.jsp.CountNum" scope="page"/>
<br>
<%-- 字体 --%>
<p><font color="blue">
<h1><jsp:getProperty name="visitcount" property="num"/></h1>
</font></p>
</body>
</html>
```

CountNum.java

```
public class CountNum {
    private int num = 0;                    //计数属性
    public int getNum() {
        num++;                              //每次访问时，自动加 1
        return num;
    }
    public void setNum(int num) {           //num 的 set 方法
        this.num = num;   }
}
```

【运行结果】运行结果如图 27-8 所示。

重新启动 Tomcat，将 JavaBean 部署到 Web 应用中，启动服务器，通过 JSP 页面上的设置可以测试不同的效果。

因为 Page 只代表目前访问的界面，所以在第一次访问 Web 应用程序时，会显示“1”，也就是当每次访问页面时，JavaBean 都会重新创建。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



图 27-8　Page 测试效果图

3．有效期 Request

当 JavaBean 的 scope 设置为“request”时，就会增加 JavaBean 的有效范围，使得 JavaBean 在有效的请求范围内有效。

4．有效期会话 session

当 JavaBean 的 scope 设置为 session 时，JavaBean 的有效范围会比前两个有所增加，此时的 JavaBean 只能在一次用户会话有效。

5．应用有效期 application

JavaBean 的 scope 设置为“application”时，JavaBean 的有效范围会比 session 广，JavaBean 可以在整个应用内有效。

6．数据库操作封装的 JavaBean

使用 JavaBean 对数据库操作进行封装，是为了隐藏 java.sql 包中的类，并且在编码中去掉了核心的数据库操作代码，这样就可以杜绝数据库操作所带来的资源释放问题，并且会相应地减少一些数据库的操作代码量。

在对数据库进行封装时，如果要返回结果集（ResultSet），那么对于封装就没有任何意义了。因为这样也是对数据库核心数据库类的操作，与封装前没有什么变化，而且结果集依赖于它使用的连接对象，因此，当连接对象在方法被关闭后，所返回的 ResultSet 就没有效果了。

如果需要获得查询数据库的结果集，可以把结果集中的数据存储到以 Map 为元素的 List 对象内，但是这种方式的缺点是不适用于大量的数据查询。以下是数据库操作 JavaBean 的简单实例。

27.8　习题

【习题 27-1】编写一个 Servlet，用来获得 get 方式提交的表单信息。

【分析】该题主要考查使用 Servlet 提交表单。

【核心代码】

```
<%@ page language="java" contentType="text/html;charset=gb2312"%>
<html>
<head>
    <title>用户表单</title>
</head>
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
<body>
    <%--表单，提交方式为 get-，提交到 DoGetDemo--%>
    <form action="DoGetDemo" method="get">
    用户名:<input type="text" name="username"/><br>
        密码:<input type="password" name="password"/><br>
        <input type="submit" value="提交"/>
        <input type="reset" value="重置"/>
    </form>
</body>
</html>
```

【习题 27-2】编写一个 Servlet，用来设置 request 属性范围，添加一个跳转页面，在该跳转页面中获得该 request 的属性范围。

【分析】该题主要考查 Servlet 之间的跳转。

【核心代码】

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    //设置输出内容格式和编码
    response.setContentType("text/html;charset=gb2312");
    PrintWriter out = response.getWriter();
    //存储在 request 范围
    request.setAttribute("name", "James");
    //取得 request 范围的 name 属性
    String name = (String) request.getAttribute("name");
```

【习题 27-3】使用 Servlet 编写一个登录界面。

【分析】该题主要考查 Servlet 中页面的编写。

【核心代码】

```
public class InitParamServlet extends HttpServlet {

    public void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        response.setCharacterEncoding("UTF-8");// 设置 response 编码方式
        request.setCharacterEncoding("UTF-8");      // 设置 request 编码方式

        response.setContentType("text/html");       // 设置类型为 HTML 类型

        PrintWriter out = response.getWriter();
        out.println("<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.01
                Transitional//EN\">");
        out.println("<HTML>");
        out.println("  <HEAD><TITLE>请登录查看 Notice 文件</TITLE></HEAD>");
        out.println("<style>body, td, div {font-size:12px; }</style>");
        out.println("  <BODY>");

        out.println("<form action='" + request.getRequestURI() + "'
                method='post'>");
      out.println("用户名：<input type='text' name='username' style='width:200px;
                '> <br/>");
        out.println("密码：<input type='password' name='password'
                    style='width:200px; '> <br/><br/>");
        out.println("<input type='submit' value='  登录  '>");
        out.println("</form>");

        out.println("  </BODY>");
        out.println("</HTML>");
        out.flush();
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
            out.close();
        }

        public void doPost(HttpServletRequest request, HttpServletResponse response)
                throws ServletException, IOException {

            String username = request.getParameter("username");// 提交的 username 参数
            String password = request.getParameter("password");// 提交的 password 参数

            Enumeration params = this.getInitParameterNames();// 所有的初始化参数名称
            while(params.hasMoreElements()){                  // 遍历所有的初始化参数
            String usernameParam = (String)params.nextElement();  // 参数名，即用户名
            String passnameParam =getInitParameter(usernameParam);// 参数值，即密码

                if(usernameParam.equalsIgnoreCase(username)
                        && passnameParam.equals(password)){ // 如果用户名和密码匹配则显示
                    request.getRequestDispatcher("/WEB-INF/notice.html").forward
(request, response);
                    return;
                }
            }
            this.doGet(request, response); // username，password 不匹配，显示登录页面
        }
    }
```

【习题 27-4】编写一个 Servlet，取得 url 中的参数值。

【分析】该题主要考查 Servlet 中页面的编写。

【核心代码】

```
    public class Servlet4 extends HttpServlet {
       public void service(HttpServletRequest req, HttpServletResponse resp) throws
IOException {
          resp.setCharacterEncoding("UTF-8");
          resp.setContentType("text/html");
          PrintWriter out = resp.getWriter();
          out.println("TEST");

          Enumeration<String> es = req.getParameterNames();
          while (es.hasMoreElements()) {
             String s = es.nextElement();
             String[] vs = req.getParameterValues(s);
             for (String a : vs) {
                out.println(s + " = " + a);
             }
          }

          out.flush();
          out.close();
       }
    }
```

【习题 27-5】编写一个 Servlet，模拟用户登录，并将用户信息放入 session。

【分析】该题主要考查 Servlet 中页面的编写。

【核心代码】

```
    public class Servlet5 extends HttpServlet {
       public void service(HttpServletRequest req, HttpServletResponse resp) throws
IOException {
          resp.setCharacterEncoding("UTF-8");
          resp.setContentType("text/html");
          PrintWriter out = resp.getWriter();
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
            User u = new User();
            u.setName(req.getParameter("name"));
            u.setPasswd(req.getParameter("passwd"));

            HttpSession session = req.getSession();
            session.setAttribute("user", u);

            User user = (User) session.getAttribute("user");
            out.println("用户名: " + user.getName());
            out.println("密码 : " + user.getPasswd());

            out.flush();
            out.close();
        }
    }
```

【习题 27-6】配置一个 Servlet，并读取配置信息。

【分析】该题主要考查 Servlet 中页面的编写。

【核心代码】

```
public class Servlet6 extends HttpServlet {
    private String message;
    private String count;
    public void init(ServletConfig config) throws ServletException {
        super.init(config);
        message = config.getInitParameter("message");
        count = config.getInitParameter("count");
    }
    public void service(HttpServletRequest req, HttpServletResponse resp) throws
IOException {
        resp.setCharacterEncoding("UTF-8");
        resp.setContentType("text/html");
        PrintWriter out = resp.getWriter();

        out.println("message: " + message + "<br/>");
        out.println("count: " + count);

        out.flush();
        out.close();
    }
}
```

【习题 27-7】编写一个 Servlet，接收表单多选框信息。

【分析】该题主要考查 Servlet 中页面的编写。

【核心代码】

```
public class Servlet7 extends HttpServlet {
    public void service(HttpServletRequest req, HttpServletResponse resp) throws
IOException {
        resp.setCharacterEncoding("UTF-8");
        resp.setContentType("text/html");
        req.setCharacterEncoding("UTF-8");
        PrintWriter out = resp.getWriter();

        out.println("<html>");
        out.println("<head><title>课程表单</title></head>");
        out.println("<body>");
        out.println("<form action='/s7' method='post'>");
        out.println("姓名: <input type='text' name='name'> <br>");
        out.println("班级: <input type='text' name='clazz'> <br>");
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
            out.println("课程：");
            out.println("<input type='checkbox' name='course' value='经济学原理'><label>
                        经济学原理</label>");
            out.println("<input type='checkbox' name='course' value='管理学原理'><label>
                        管理学原理</label>");
            out.println("<input type='checkbox' name='course' value='营销学原理'><label>
                        营销学原理</label>");
            out.println("<input type='checkbox' name='course' value='会计学原理'><label>
                        会计学原理</label>");
            out.println("<br>");
            out.println("<input type='submit' value='提交'>");
            out.println("</form>");

            out.println("==========================<br/>");

            String name = req.getParameter("name");
            if (!(null == name || "".endsWith(name))) {
               String clazz = req.getParameter("clazz");
               String[] courses = req.getParameterValues("course");

               out.println("你提交的表单信息<br>");
               out.println("姓名："+name+"<br>");
               out.println("班级："+clazz+"<br>");
               out.println("课程：<br/>");
               if (null != courses) {
                  for (String c : courses) {
                     out.println("   " + c + "<br>");
                  }
               }
            }
            out.println("</body>");
            out.println("</html>");

            out.flush();
            out.close();
         }
      }
```

【习题 27-8】编写一个 Servlet，跳转到另一个 Servlet，然后显示 url 参数信息。

【分析】该题主要考查 Servlet 中页面的编写。

【核心代码】

```
public class Servlet8 extends HttpServlet {
   public void service(HttpServletRequest req, HttpServletResponse resp) throws
IOException, ServletException {
      resp.setCharacterEncoding("UTF-8");
      resp.setContentType("text/html");
      req.setCharacterEncoding("UTF-8");

      RequestDispatcher d = req.getRequestDispatcher("/s7");
      d.forward(req, resp);
   }
}
```

【习题 27-9】编写一个 Servlet，重定向到另一个 Servlet，然后读取出 url 中的参数值。

【分析】该题主要考查 Servlet 中页面的编写。

【核心代码】

```
public class Servlet9 extends HttpServlet {
   public void service(HttpServletRequest req, HttpServletResponse resp) throws
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
IOException, ServletException {
            resp.setCharacterEncoding("UTF-8");
            resp.setContentType("text/html");
            req.setCharacterEncoding("UTF-8");

            resp.sendRedirect("http://www.baidu.com");
        }
    }
```

【习题 27-10】编写一个 Servlet，将用户信息写入 cookie。

【分析】该题主要考查 Servlet 中页面的编写。

【核心代码】

```
    public class Servlet10 extends HttpServlet {
        public void service(HttpServletRequest req, HttpServletResponse resp) throws
IOException, ServletException {
            resp.setCharacterEncoding("UTF-8");
            resp.setContentType("text/html");
            req.setCharacterEncoding("UTF-8");

            User u = new User();
            u.setName(req.getParameter("name"));
            u.setPasswd(req.getParameter("passwd"));

            Cookie c = new Cookie("user", u.getName() + "/"+ u.getPasswd());
            c.setMaxAge(-1);
            c.setSecure(true);
            resp.addCookie(c);
        }
    }
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第 4 篇　综合案例篇

第 28 章　智达图书馆管理系统

随着人类社会的不断进步，科学技术有了长足的发展，尤其是在当今这个知识经济时代，知识的大爆炸带来全球图书业的飞速发展，这些书籍为科技的发展做出了巨大的贡献。

人类在短短的几十年创造了大量的书籍资源，但图书数量之多，也使其管理成为一项繁重的工作。随着信息时代的来临，图书的信息化管理使这个难题的解决成为现实。本章将通过具体实例的介绍，引领读者亲身体验“图书馆管理系统”的开发之旅。



28.1　项目立项

系统不但要满足客户所提出的要求，同时也要考虑其性能。因此，在项目开发之前，首先要进行需求分析和可行性分析，这是进行项目开发的基石。

28.1.1　需求分析

通过与图书管理人员进行详细的交流，最终确定本系统要具备以下功能。

1．学生用户端

- 查询图书：学生用户可以进行简单查询和高级查询，后面将会对其进行详细讲解。
- 预约图书：当要借的书不在馆，可以提前预约。
- 挂失图书：图书丢失需要挂失，可在学生端实现。

2．管理员端

管理端的功能比较复杂，如管理学生、图书和管理员等信息，还需要对借阅信息进行管理，具体功能如下。

- 学生用户管理：实现学生用户信息的增、删、改、查。
- 图书管理：包括对图书的增、删、改等。
- 管理员管理：操作者包括超级管理员和普通管理员，超级管理员可对任意管理员的信息进行增、删、改、查，而普通管理员只有修改自己密码的权限。
- 借阅管理：主要是学生借阅图书、归还图书和缴纳罚款等的管理。

28.1.2　可行性分析

可行性分析是对项目的可行程度进行分析，以便管理层对资金及技术的投入进行决策，具体包括技术可行性、经济可行性、营运可行性。通过对需求分析的研究，对本项目的可行性分析如下。

1．引言

图书是人类获取知识最重要的途径之一。随着读者的日益增多，对图书的管理要求也日益提高，因此，实现图书的信息化管理势在必行。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

（1）编写目的

从图书馆的需求及现有技术等方面研究该图书管理系统的可行性，为公司上层进行决策提供依据。

（2）项目背景

- 项目提出者：某学校图书馆。
- 项目开发者：某软件公司。
- 系统用户：该校众多学生用户及图书管理人员。

2. 可行性分析的前提

（1）系统要求

学生端需要包含学生借阅图书基本信息、预约图书基本信息、挂失图书基本信息、超期图书基本信息，而管理员端除了上述四项信息外，还包含学生基本信息、图书基本信息、管理员基本信息。所有的管理员均可以对其包含的信息进行基本的查询。

（2）系统目标

图书馆管理系统的目标是：信息收集制度化、信息加工标准化、信息存储档案化、信息传递规范化及图书馆信息工作专业化。

3. 对图书馆支出与收益的分析

当前该图书馆的管理比较粗略，还有大部分工作需要手工操作，对信息的维护不仅费用高、工作量大，而且对信息的安全性很难得到保障。

学校支出了适量的资金进行本系统的开发。使用本系统后，大部分工作实现了自动化，图书馆只花费很少的人力和物力进行系统的维护即可，并降低了数据被无意破坏的风险。

4. 对公司技术支持的分析

该系统可以采用 Swing+MySQL 数据库来完成，公司之前已经开发过很多类似的项目，完全可在要求的时间内完成该系统的开发。

5. 对公司支出与收益的分析

该项目要求在 20 天内完成，要求有一年的维护期，经分析，去除开发费用及维护费用，公司最后所得利润为 1 万元。对该系统的开发既可以增加公司的技术储备，又可增加对类似项目开发的熟练程度。

6. 社会因素的分析

该系统完全由本公司独立开发完成，是按本公司的开发体系结构进行开发，在法律方面没有任何侵权行为，完全符合合同的规定。

7. 结论

根据上述分析，公司认为此项目结构清晰、功能合理，而且使用 Java SE 技术能够进行开发，并实现客户要求的全部功能。由于这是一个中小型系统，客户要求的开发时间完全充裕，利润与开发成本比比较高，这可在一定程度上提高公司的效益。因此，公司决定开发此项目。

28.2 详细设计

通过与图书馆管理人员交流并进行可行性分析，公司对本系统的功能及性能要求有了较深

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

入的了解，下面就可以确定系统的最终目标及初步的功能结构。

28.2.1　系统目标

通过系统分析，并与图书馆管理人员再次探讨，最终确定系统的最终目标如下。

- 实现需求分析阶段客户提出的功能。
- 提高键盘易用性，减少鼠标操作。
- 使用当前比较流行的树状列表控件作为导航，直观地显示系统结构。

28.2.2　系统功能结构

本系统包括学生端和管理员端，有借阅图书、预约图书、查询图书、挂失图书、超期交纳罚金等功能模块，其系统结构如图 28-1 所示。

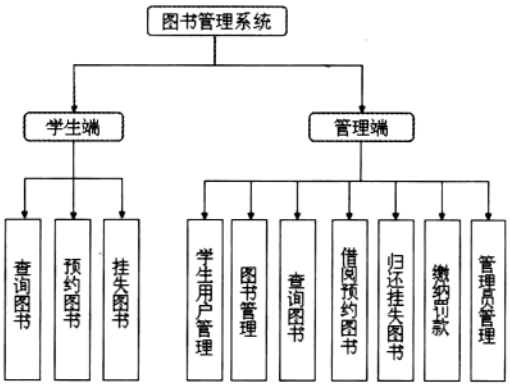


图 28-1　系统结构图

28.2.3　构建开发环境

开发此图书管理系统需要用到的软件环境如下。

- JDK 1.6 及其以上版本：1.6 版本作为目前最新版本，增加了许多新特性，读者可到 www.sun.com 网站免费下载。
- 数据库：MySQL 5.0 及其以上版本。MySQL 是一款免费软件，大大降低了中小型系统的开发成本，功能比较全面。因此，本系统选择使用 MySQL 数据库。

28.2.4　学生端系统预览

学生端由四个功能模块组成，其主要操作如下：

① 进入登录界面，输入各项信息，单击“学生登录”按钮，如果输入信息无误，用户就可以进入学生管理系统主界面。如果学生密码或用户名输入错误，将弹出提示对话框。该登录界面如图 28-2 所示。

② 学生用户从图 28-2 登录成功后，就进入学生管理系统，在这里展开左边的树状列表后，可以通过单击不同节点进行相应的操作。当用户单击“退出”节点时，弹出选择对话框，供用户选择是否退出系统，其界面如图 28-3 所示。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

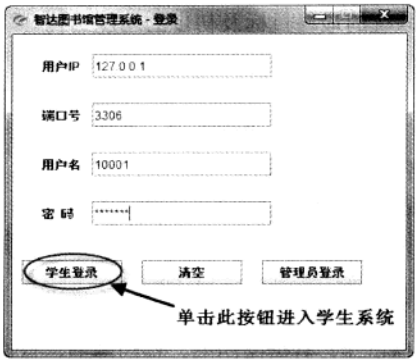


图 28-2　登录界面

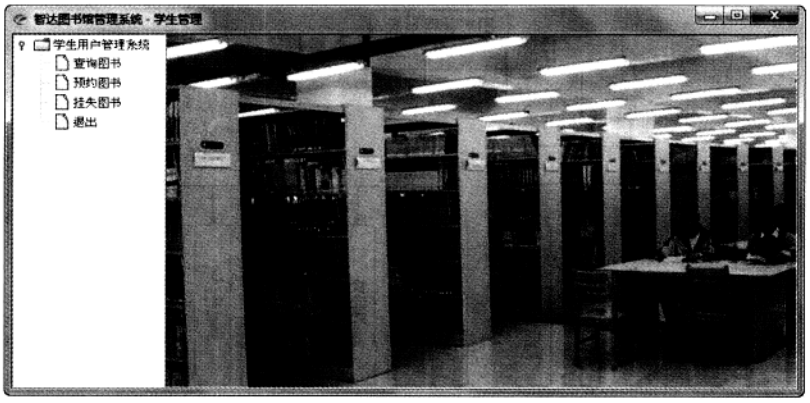


图 28-3　学生管理系统界面

③ 学生用户进入管理系统后，选中左边树状列表中的“查询图书”节点便可进行查询操作。例如，选中“简单查询”单选按钮，并在下拉列表框中选择查询条件，然后在其后面的文本框中输入要查询的内容即可查询。如果查询结果不为空，就显示在界面右下部分的表格中，其效果如图 28-4 所示。

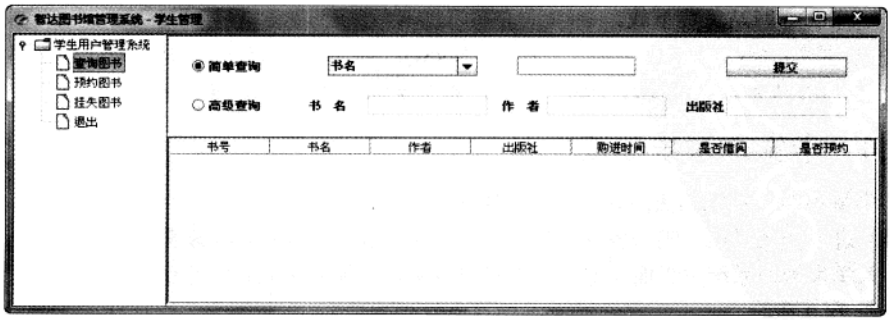


图 28-4　查询图书

④ 选中“预约图书”节点，就进入其对应的功能界面。选择“借阅图书”单选按钮，并在“学号”和“书号”文本框输入正确的内容，单击“确定”按钮即可执行需借阅图书的查询，效果如图 28-5 所示。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

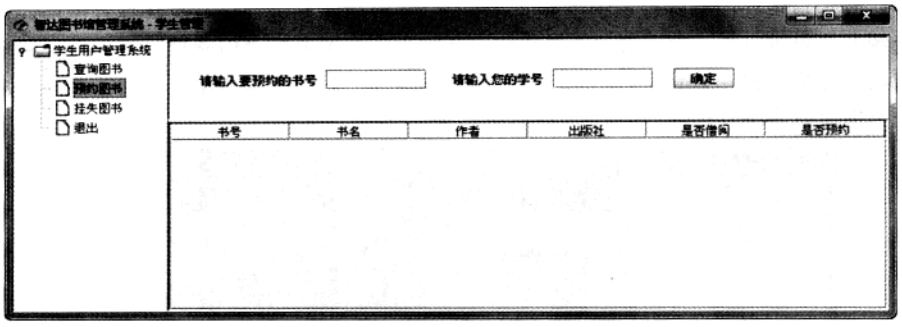


图 28-5　预约图书

⑤ 选中“挂失图书”节点后，首先在“学号”文本框中输入学号，单击“归还”按钮，用户的借书信息将显示在界面右下部分的表格中。当选中某一行图书信息后单击“确定”按钮，即可完成挂失图书功能，如图 28-6 所示。

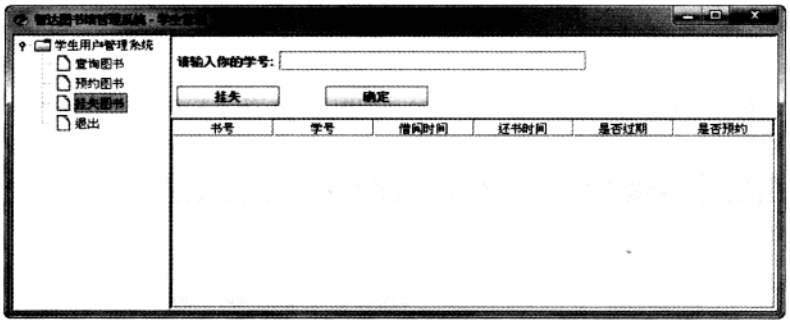


图 28-6　挂失图书

28.2.5　管理员端系统预览

前面介绍了学生用户端各界面的预览及其简单操作，下面将对管理员端各界面进行介绍，具体的各界面效果图及简单操作步骤如下。

① 管理员进行系统管理前，首先需要进入登录界面。当管理员在该界面输入正确的信息后，单击“管理员登录”按钮就可进入图书管理系统界面，登录界面如图 28-7 所示。

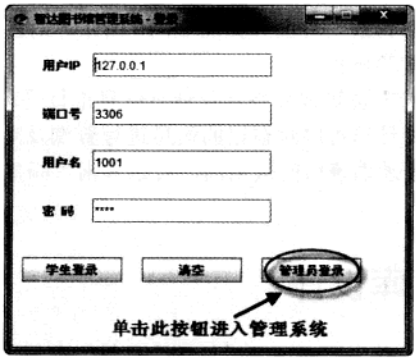


图 28-7　管理员登录窗口

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

② 管理员可在图书管理系统主界面（如图 28-8 所示）进行查看及管理等操作，也可对学生及图书信息进行增、删、改、查等操作。其中“查询图书”、“借阅预约图书”、“归还挂失图书”及“交纳欠款”子节点的功能与学生管理系统相同，在此不再重复介绍，其他节点的功能将在下面进行详细介绍。



图 28-8　图书管理系统界面

③ 管理员可以在“学生用户管理”界面（如图 28-9 所示）对学生用户信息进行管理，包括对其进行添加、删除、修改、查询等操作。

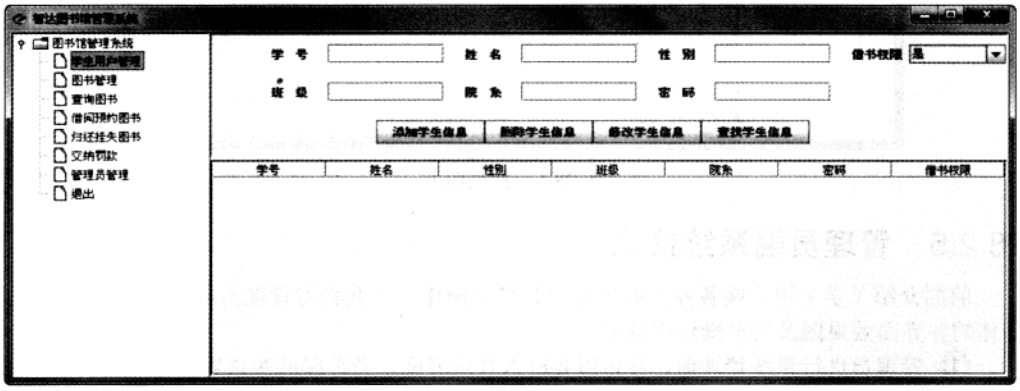


图 28-9　学生用户管理界面

④ 管理员可以在“图书管理”界面（如图 28-10 所示）对图书信息进行管理，包括对其进行添加、删除、修改、查询等操作。

⑤ 超级管理员可以在管理员管理界面对管理员信息进行管理，包括对其进行添加、删除、修改、查询等操作；普通管理员只可以对自己的信息进行查询及对自己的密码进行重置。如图 28-11 所示的是普通管理员登录系统后的效果图，可以看出“添加管理员”、“删除管理员”两个按钮为不可编辑状态。

28.3　数据库设计

数据库作为系统的基础，首先要保证其设计的合理性。在使用应用系统时，拥有设计合理的数据库往往可以起到事半功倍的效果。在系统的后续维护、功能扩展中，这一优点体现得更

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

是淋漓尽致。

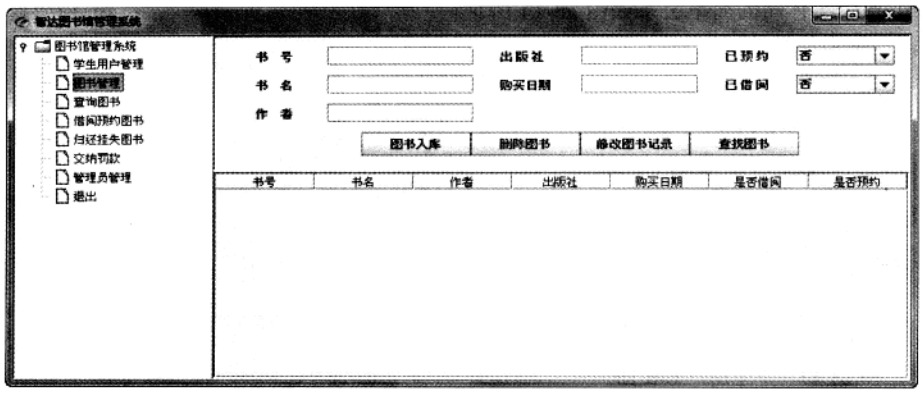


图 28-10　图书管理界面

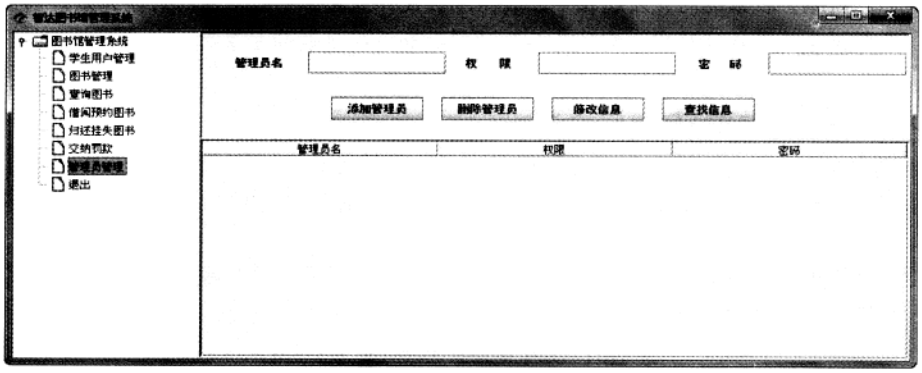


图 28-11　管理员管理界面

28.3.1　数据库分析

由于系统规模比较小，对数据库的要求不算高，故采用适合中小型企业使用的 MySQL 作为系统开发的数据库，而 MySQL 从 5.0 版开始支持事务，这对数据的完整性有了很好的保障。

28.3.2　E-R 图

该系统包含的实体主要有：图书、学生、管理员等，下面将分别介绍各实体及实体间的 E-R 图。通过其 E-R 图的介绍，读者可以更好地理解各实体的属性关系。

① 管理员和图书的 E-R 图如图 28-12 所示。

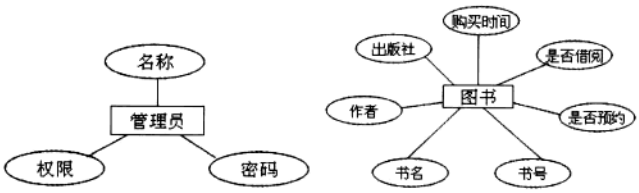


图 28-12　管理员、图书 E-R 图

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

② 学生 E-R 图如图 28-13 所示。

③ 各个实体间的简单关系如图 28-14 所示。

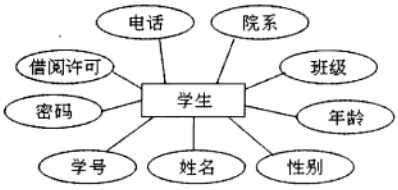


图 28-13 学生用户 E-R 图

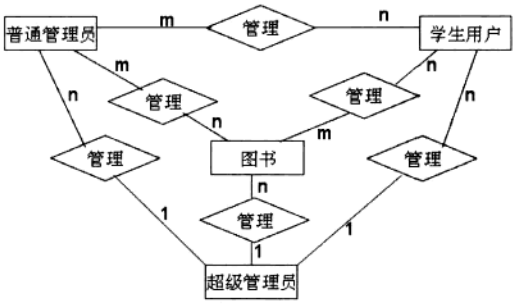


图 28-14 实体间简单关系图

说明：在系统的开发过程中，数据库的设计直接影响着系统的后期开发。因此，在数据库的设计过程中，需要为后期的维护和系统性能的扩展打下良好的基础。

28.3.3 数据表设计

本系统共有七张表：学生用户基本信息表、管理员基本信息表、图书基本信息表、借阅图书信息表、超期图书信息表、预约图书信息表、挂失图书信息表，下面将分别对各表进行介绍。

① 学生用户基本信息表：用于记录学生用户的基本信息，其主要字段为学生用户名和密码等，具体设计如表 28-1 所示。

表 28-1 学生用户基本信息表 student

字段名称	数据类型	字段大小	是否主键	是否为空	说　明
StuNO	int	N/A	是	否	学号
StuName	varchar	50	否	是	姓名
StuAge	int	N/A	否	是	年龄
StuSex	varchar	50	否	是	性别
Class	varchar	50	否	是	班级
Department	varchar	50	否	是	院系
Tel	char	11	否	是	电话
Permitted	varchar	50	否	是	借阅许可
Password	varchar	20	否	是	密码

建立本表的 SQL 语句如下：

```
    create table student(StuNO int primary key,StuName varchar(50),StuAge int,StuSex
varchar(50),
    Class varchar(50),Department varchar(50),Tel char(11),Permitted varchar(50),
Password varchar(20));
```

② 管理员基本信息表：用于记录管理员的用户 ID、密码及权限，详细情况如表 28-2 所示。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

表 28-2　管理员基本信息表 manager

字段名称	数据类型	字段大小	是否主键	是否为空	说　明
mgNo	int	N/A	是	否	管理员 ID
permitted	varchar	50	否	否	管理员权限
password	varchar	50	否	否	管理员密码

建立本表的 SQL 语句如下：

```
creat table manager(mgNo int primary key,permitted varchar(50),password
varchar(50));
```

③ 图书基本信息表：该表用于记录图书的基本信息，具体实现如表 28-3 所示。

表 28-3　图书基本信息表 book

字段名称	数据类型	字段大小	是否主键	是否为空	说　明
BookNO	int	N/A	是	否	图书 ID
BookName	varchar	50	否	是	图书名称
Author	varchar	50	否	是	图书作者
Publishment	varchar	50	否	是	出版社
ButTime	varchar	50	否	是	购进日期
Borrowed	varchar	50	否	是	是否借阅
Ordered	varchar	50	否	是	是否预约

建立本表的 SQL 语句如下：

```
create table book(BookNO int primary key,BookName varchar(50),Author varchar(50),
Publishment varchar(50),ButTime varchar(50),Borrowed varchar(50),Ordered
varchar(50));
```

④ 借阅图书信息表：该表用来记录学生用户借阅图书的基本信息，包括学号、书号、借阅时间等信息，各字段的详细信息如表 28-4 所示。

表 28-4　借阅图书信息表 record

字段名称	数据类型	字段大小	是否主键	是否为空	说　明
BookNO	int	N/A	是	否	书号
StuNO	Int	N/A	否	否	学号
BorrowTime	Varchar	50	否	是	借阅时间
ReturnTime	varchar	50	否	是	归还时间
Borrowed	varchar	50	否	是	是否借阅
Ordered	varchar	50	否	是	是否预约

建立本表的 SQL 语句如下：

```
create table record(BookNO int primary key,StuNO int,BorrowTime varchar(50),
ReturnTime varchar(50),Borrowed varchar(50),Ordered varchar(50));
```

⑤ 超期图书信息表：该表用来记录学生超期图书的信息，包括图书 ID 和学生姓名等信息，具体设计如表 28-5 所示。

表 28-5　图书超期信息表 exceedTime

字段名称	数据类型	字段大小	是否主键	是否为空	说　明
StuNO	int	N/A	是	否	学号
BookNO	int	N/A	是	否	书号

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

续表

字段名称	数据类型	字段大小	是否主键	是否为空	说　明
BookName	Varchar	50	否	是	书名
DelayTime	Varchar	50	否	是	超期天数

建立本表的 SQL 语句如下：

```
create table exceedtime(StuNO int,BookNO int,BookName varchar(50),DelayTime int,
primary key(StuNO,BookName));
```

⑥ 预约图书信息表：该表用来记录学生预约的图书信息，包括图书 ID 和学生姓名等信息，具体设计如表 28-6 所示。

表 28-6　预约图书信息表 orderreport

字段名称	数据类型	字段大小	是否主键	是否为空	说　明
BookNO	Int	N/A	是	否	书号
StuName	varchar	50	否	是	学生姓名
Class	varchar	50	否	是	班级
BookName	varchar	50	否	是	书名
StuNO	Int	N/A	否	是	学生 ID
Author	varchar	50	否	是	作者

建立本表的 SQL 语句如下：

```
create table orderreport(BookNO int primary key,StuName varchar(50), Class
varchar(50),
BookName varchar(50),StuNO int,Author varchar(50));
```

⑦ 挂失图书信息表：该表用来记录学生丢失图书的信息，包括图书 ID 和学生姓名等信息，具体设计如表 28-7 所示。

表 28-7　挂失图书信息表 losebook

字段名称	数据类型	字段大小	是否主键	是否为空	说　明
LBNO	int	N/A	是	否	挂失书号
BookNO	int	N/A	否	是	书号
BookName	Varchar	50	否	是	书名
StuNO	int	N/A	否	是	学生 ID

建立本表的 SQL 语句如下：

```
create table losebook(LBNO int primary key,StuNO int,BookNO int,BookName
varchar(50));
```

28.3.4　导入数据

前面已经对表的设计进行了详细的介绍，下面将介绍如何在 MySQL 数据库中创建这些表，并插入一些数据用于开发中的测试，具体步骤如下：

① 启动 MySQL，打开 MySQL 命令行客户端 MySQL Command Line Client。

② 执行语句“use test;”进入数据库 test，开始对其进行操作。

③ 执行上一节介绍的创建表的 SQL 语句。

表创建完成后，便可输入一些测试数据了，插入测试数据的 SQL 语句如下。

```
insert into book values(10001,'oracle 快速入门','王海亮','水利出版社','2003.1.12','
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
否','否');
    insert into book values(10002,'数据库项目案例','周兴华','清华出版社','2003.3.15','否
','否');
    ……
```

对于各张表中信息的插入，请参考光盘中本章的数据库脚本 db.sql，在此就不一一列举了。至此，代码开发前的准备工作已经完成，在后面将对系统中的各模块按照实际的开发流程逐一进行介绍。

28.4　数据库接口

在开发具体的功能之前，做好数据库操作类的开发，主要是用于连接数据库、执行查询、更新等功能，具体实现代码如下。

代码位置：见光盘中本章源码的 DataBase.java 类。

```
1   public class DataBase{
2       Connection con=null;                    //声明 Connection 引用
3       Statement stat;                         //声明语句引用
4       ResultSet rs;                           //声明结果集引用
5       int count;                              //用于记录更新记录数
6       public static String message;           //声明一个静态成员变量
7
8       public DataBase(){                      //构造器
9           try{
10            Class.forName("com.mysql.jdbc.Driver");   //加载驱动
11              con=DriverManager.getConnection("jdbc:mysql://"+message+"
                /test","root","");                      //创建连接
12              stat=con.createStatement();             //创建 Statement 对象
13          }
14          catch(Exception e){             //如果从 Login 类传的参数不对，则提示出错
15          JOptionPane.showMessageDialog(null, "用户 IP 或端口号错误!!! ",
16                      "信息",JOptionPane.INFORMATION_MESSAGE);   //弹出信息提示框
17      }}
18      public void selectDb(String sql){  //数据库查询方法
19          try{rs=stat.executeQuery(sql); }            //执行查询
20          catch(Exception ie){ie.printStackTrace();}  //捕获异常并打印
21      }
22      public int updateDb(String sql){                //声明 update 方法
23          try{
24              sql = new String(sql.getBytes(),"ISO-8859-1");
                                            //将编码格式转为 ISO-8859-1
25              count=stat.executeUpdate(sql);          //执行更新
26          }
27          catch(Exception ie){ie.printStackTrace();}  //打印异常
28          return count;                               //返回更新记录数
29      }
30      public void dbClose(){                          //声明 close 方法
31          try{con.close();}                           //关闭连接
32          catch(Exception e){e.printStackTrace();}    //打印异常
33      }}
```

- 第 10～12 行代码主要是加载 MySQL 数据库的 JDBC 驱动类，并创建数据库连接和 statement 对象。
- 第 14～17 行代码是对数据库连接时接收了错误参数进行的错误信息提示。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

- 第 18～33 行代码，分别为数据库查询方法、更新方法及关闭方法。这几个方法在本系统开发时会经常被调用。

说明：代码第 6 行声明了一个静态成员变量，其值是用户在登录时给出的，在本地机器上进行测试时，可以用“127.0.0.1:3306”进行测试。

28.5 登录模块



图 28-15 登录界面

用户登录界面的主要功能是验证登录者的身份和权限，只有学生用户或管理员才能登录与之对应的管理系统。在登录界面，通过对各项登录条件的验证，就可以确认用户的使用权限。下面将介绍登录模块的具体开发，登录界面的运行效果如图 28-15 所示。

28.5.1 登录界面

开发登录模块的具体功能前，首先要完成其界面的搭建，然后开发各功能的具体实现。下面先介绍界面的搭建过程，代码如下。

代码位置：见光盘中本章源代码的 Login.java 类。

```
1   public class Login extends JFrame implements ActionListener{
2      private JPanel jp=new JPanel();                          //创建 JPanel 对象
3      private JLabel []jlArray={                               //创建标签组
4          new JLabel("用户 IP"),new JLabel("端口号"),new JLabel("用户名"),
5          new JLabel("密  码"),new JLabel("")                  //初始化标签
6      };
7      private JButton[] jbArray={                              //创建按钮数组
8       new JButton("学生登录"),new JButton("清空"),new JButton("管理员登录")
                                                                //初始化按钮数组
9      };
10     private JTextField[] jtxtArray={                         //创建文本框
11         new JTextField("127.0.0.1"),new JTextField("3306"),new JtextField
           ("1001")                                             //初始化文本框数组
12     };
13     private JPasswordField jpassword=new JPasswordField("1001"); //创建密码框
14     String sql;                                              //用于存放 sql 语句
15     DataBase db;                                             //声明数据库类引用
16  }
```

说明：以上代码只是对登录界面所需控件的初始化，创建了一系列的标签、按钮、输入文本框。

下面在 Login 类的构造函数中对各控件的大小、位置进行设置及注册事件监听器，对标题和 logo 图片进行初始化，并设置窗体的大小、位置及可见性。其代码如下。

代码位置：见光盘中本章源代码的 Login.java 类。

```
1   public Login(){
2         jp.setLayout(null);                                   //设置 JPanel 的布局管理器
3         for(int i=0;i<4;i++){                                 //对标签与按钮控件循环处理
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
4            jlArray[i].setBounds(30,20+i*50,80,25); //设置标签与按钮的大小和位置
5            jp.add(jlArray[i]);          //将标签和按钮添加进 JPanel 容器中
6         }
7         for(int i=0;i<3;i++){           //设置按钮的大小、位置，并为其添加事件监听器
8            jbArray[i].setBounds(10+i*120,230,100,25);
9            jp.add(jbArray[i]);                        //将按钮添加到面板中
10           jbArray[i].addActionListener(this);        //注册动作事件监听器
11        }
12        for(int i=0;i<3;i++){           //设置文本框的大小、位置，并注册事件监听器
13           jtxtArray[i].setBounds(80,20+50*i,180,25);
14           jp.add(jtxtArray[i]);                      //将文本框添加到面板中
15           jtxtArray[i].addActionListener(this);      //注册动作事件监听器
16        }
17        jpassword.setBounds(80,170,180,25);           //设置密码框的大小位置
18        jp.add(jpassword);                            //将密码框添加进 JPanel 容器
19        jpassword.setEchoChar('*');                   //设置密码框的回显字符
20        jpassword.addActionListener(this);            //为密码框注册监听器
21        jlArray[4].setBounds(10,280,300,25);//设置用于显示登录状态的标签的大小，位置
22        jp.add(jlArray[4]);                           //将标签添加进 JPanel 容器
23        this.add(jp);                                 //将面板添加到窗体中
24        Image image=new ImageIcon(this.getClass().getResource("/ico.png")).
          getImage();                                   //对 logo 初始化
25        this.setIconImage(image);                     //设置图标
26        this.setTitle("智达图书馆管理系统 - 登录");     //设置窗口标题
27        this.setResizable(false);                     //使最大化按钮不可用
28        this.setBounds(100,100,400,350);                        //设置窗口大小及位置
29        this.setVisible(true);                                  //设置窗口可见性
30        this.addWindowListener(new WindowAdapter() {            //定义窗口关闭监听器
31           public void windowClosing(WindowEvent e) {
32              System.exit(0);
33           }
34        });
35     }
```



说明：关于登录模块的其他功能代码，将逐渐添加进 Login 类体，另外，各控件监听代码的开发将在以后的模块中详细介绍。

28.5.2　键盘易用性

管理系统使用频繁，输入工作量巨大，因此，键盘易用性的问题必须考虑。本节将介绍图书馆管理系统中键盘易用性的开发，其代码如下。



提示：所谓键盘易用性，一般是指在需要输入很多不同信息的界面中在一种信息输入完成后，按 Enter 键，输入焦点自动跳转到下一个要输入信息的控件。对于本系统而言，既可以方便学生用户的操作，又可以提高管理员的工作效率。

代码位置：见光盘中本章源代码的 Login.java 类。

```
1   public void actionPerformed(ActionEvent e)
2   {                                                  //得到登录相关信息
3       String mgno=jtxtArray[2].getText().trim();     //得到登录名
4       String mgIP=jtxtArray[0].getText().trim();     //得到 IP 地址
5       String port=jtxtArray[1].getText().trim();     //得到端口号
6       String message=mgIP+":"+port;
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
7        DataBase.message=message;//将本方法的变量作为参数赋值给 DataBase 方法的私有变量
8        DataBase.log=this;
9        if(e.getSource()==jtxtArray[0]){                    //事件源为用户 IP 文本框
10          jtxtArray[1].requestFocus();                     //切换输入焦点到端口号文本框
11           }
12           if(e.getSource()==jtxtArray[1]){                //事件源为端口文本框
13              jtxtArray[2].requestFocus();                 //切换输入焦点到用户名文本框
14           }
15           if(e.getSource()==jtxtArray[2]){                //事件源为用户名文本框
16              jpassword.requestFocus();                    //切换输入焦点到密码框
17           }
18           else if(e.getSource()==jbArray[1]){             //事件源为清空按钮
19            jlArray[4].setText(""); jtxtArray[2].setText("");jpassword.
              setText("");                                   //清空所有的信息
20            jtxtArray[2].requestFocus();                   //将输入焦点设置到 jtxtArray[2]
21            }
22   }
```

- 第 9～17 行代码实现了键盘易用性。例如，事件源为用户 IP 文本框，当按 Enter 键时，鼠标焦点到达端口号文本框，用户就可以直接在此输入端口号。
- 第 18～21 行代码是监听“清空”按钮的代码。当单击“清空”按钮时，输入文本框会全部被清空。

28.5.3 管理员登录

28.5.2 节实现了键盘易用性功能，本节将为“管理员登录”按钮编写监听代码，实现单击“管理员登录”按钮就可进入图书管理系统主界面的功能。具体代码如下。

代码位置：见光盘中本章源代码的 Login.java 类。

```
1    else if(e.getSource()==jbArray[2]){                     //事件源为管理员登录按钮
2       if(!mgno.matches("\\d+")){                           //用户名格式错误
3        JOptionPane.showMessageDialog(this,"用户名格式错误!!! ","信息",
4                                 JOptionPane.INFORMATION_MESSAGE); //弹出信息提示对话框
5        return;                                             //返回
6       }
7       if(jtxtArray[0].getText().trim().equals("")){        //如果用户 IP 为空
8        JOptionPane.showMessageDialog(this,"用户 IP 不能为空!!! ","信息",
9                                    JOptionPane.INFORMATION_MESSAGE);
                                                             //弹出信息提示对话框
10       return;                                             //返回
11      }
12      if(jtxtArray[1].getText().trim().equals("")){        //如果端口号为空
13          JOptionPane.showMessageDialog(this,"用户端口号不能为空!!! ","信息",
14                     JOptionPane.INFORMATION_MESSAGE);     //弹出信息提示对话框
15          return;                                          //返回
16          }
17      sql="select mgNo,password from manager where mgNo="+Integer.parseInt
        (mgno);                                              //搜索信息的 SQL
18      db=new DataBase();                                   //创建 DataBase 对象
19      db.selectDb(sql);                                    //进行查询，验证身份
20      try{
21            String mgNo="";                                //声明用户名
22            String password="";                            //声明密码
23            while(db.rs.next()){                           //取出结果集中数据并赋值
24                mgNo=db.rs.getString(1).trim();            //得到用户名
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



```
25              password=db.rs.getString(2).trim();      //得到密码
26      }
27      if(jtxtArray[2].getText().trim().equals(mgNo)&&
28          String.valueOf(jpassword.getPassword()).equals(password)){//登录成功
29            jlArray[4].setText("恭喜您，登录成功!!! ");      //调置 jlArray[4]的值
30          new Root(mgNo);                                  //创建主界面
31       this.dispose();                                     //将登陆窗体隐藏
32          }
33      else{                                                //登录失败
34           jlArray[4].setText("对不起，登录失败!!! ");      //设置提示信息
35          }}
36           catch(Exception e1){e1.printStackTrace();}     //捕获异常并打印
37       db.dbClose();                                       //关闭数据库链接
38   }
```

- 第 2～16 行代码对输入信息的合法性进行了验证。第 2～7 行判断用户名格式是否正确，如输入格式错误的用户名，将弹出提示对话框。第 8～11 行、第 12～16 行分别对“用户 IP”和“端口号”文本框输入为空进行了处理。
- 第 17 行的 SQL 语句的作用是从数据库中查询与用户输入的信息相匹配的信息，搜索出结果就提示登录成功，否则提示登录失败。

28.5.4　学生登录

28.5.3 节编写了管理员登录事件监听的监听代码，本节将开发学生登录动作的监听代码，实现单击“学生登录”按钮就可进入学生管理系统主界面的功能。具体代码如下。

```
1   else if(e.getSource()==jbArray[0]){          //事件源为学生登录按钮
2       if(!jtxtArray[2].getText().trim().matches("\\d+")){
                                                 //若学号格式错误，输出提示对话框
3           JOptionPane.showMessageDialog(this,"输入有误,学号只能为数字!!!",
4                               "消息", JOptionPane.INFORMATION_MESSAGE);
                                                 //弹出信息提示对话框
5           return;                              //返回
6       }
7       if(jtxtArray[0].getText().trim().equals("")){
                                                 //若“用户 IP”文本框输入为空，提示
8       JOptionPane.showMessageDialog(this,"用户 IP 不能为空!!! ","信息",
9                               JOptionPane.INFORMATION_MESSAGE);
                                                 //弹出信息提示对话框
10      return;                                  //返回
11      }
12      if(jtxtArray[1].getText().trim().equals("")){ //若“端口号”文本框输入为空，提示
13          JOptionPane.showMessageDialog(this,"用户端口号不能为空!!! ",
14                      "信息",JOptionPane.INFORMATION_MESSAGE);
                                                 //弹出信息提示对话框
15          return;                              //返回
16      }
17      //查询学号文本中所输学号是否存在于 STUDENT 表中
18      sql="select StuNO,Password from STUDENT where StuNO="
19                          +Integer.parseInt(jtxtArray[2]. getText().trim());
20      db=new DataBase();                       //创建 DataBase 类对象
21      db.selectDb(sql);                        //执行查询
22      try{
23          if(!(db.rs.next())){                 //若学号错误，输出提示对话框
24              JOptionPane.showMessageDialog(this,"输入了错误的学号!! ",
25                      "消息", JOptionPane.INFORMATION_MESSAGE);
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
                                                          //弹出信息提示对话框
26          }
27          else{String stuNO=db.rs.getString(1).trim();          //得到学号
28              password=db.rs.getString(2).trim();               //得到密码
29              if(jtxtArray[2].getText().trim().equals(stuNO)&&
30              String.valueOf(jpassword.getPassword()).equals(password)){
                                                                  //登录成功
31              jlArray[4].setText("恭喜您，登录成功!!! ");          //给出提示消息
32              new StudentSystem();                              //创建学生端窗体
33              this.dispose();                                   //将登录窗体隐藏
34          }
35          else{                                                 //登录失败
36              jlArray[4].setText("对不起，登录失败!!! ");          //显示提示信息
37          }}}
38      catch(Exception ex) { ex.printStackTrace();}              //捕获异常并打印
39       db.dbClose();                                            //关闭数据库链接
40  }
```

提示：本节必须对结果集进行异常处理，在使用完后注意关闭数据库，否则将会影响系统效率。

至此，用户登录模块的开发已经基本完成，运行 Login 的 main()方法进行测试，结果如图 28-2 所示。

28.6 管理模块

28.5 节是对登录模块的开发，本节将带领读者进行管理端的开发，具体实现见下面的内容，管理端开发完毕的运行效果如图 28-16 所示。

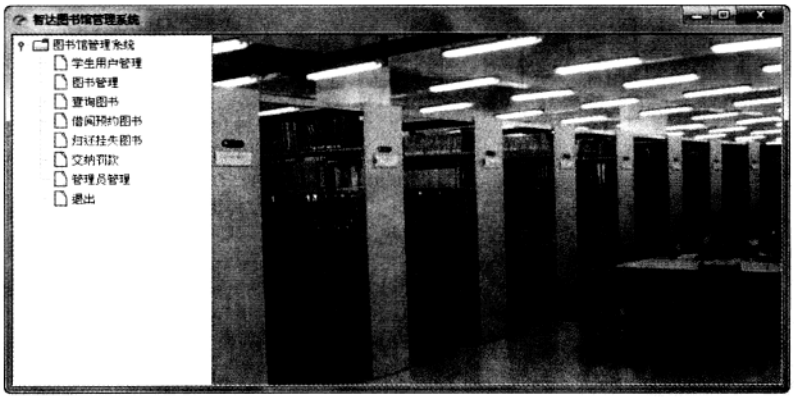


图 28-16 图书管理系统界面

28.6.1 管理端界面的初始化

本节实现管理端界面的初始化，窗体的左边是一个树状列表控件，右边是一个卡片布局的面板，这种结构在当前市面上比较流行，其实现代码如下。

代码位置：见光盘中本章源代码的 Root.java 类。

```
1   public class Root extends JFrame{
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



```
2        DefaultMutableTreeNode[] dmtn=      //创建节点数组
3                {new DefaultMutableTreeNode(new NodeValue("图书馆管理系统")),
4                ……//此处省略了创建其他树节点的代码
5                 new DefaultMutableTreeNode(new NodeValue("退出"))};
                                             //初始化树节点树组
6       DefaultTreeModel dtm=new DefaultTreeModel(dmtn[0]);
                                             //创建树模型指定根节点为“学生管理系统”
7       JTree jt=new JTree(dtm);             //创建包含 dtm 树模型的树对象
8       JScrollPane jsp=new JScrollPane(jt);//为 JTree 创建滚动窗体
9       private JSplitPane jsplr=new JSplitPane(JSplitPane.HORIZONTAL_SPLIT,
        true);                               //创建分割窗体对象
10      private JPanel jp=new JPanel();      //创建 JPanel 对象
11      Image image=new ImageIcon(this.getClass().getResource("/foreface.jpg")).
        getImage();                          //Image 对象
12       ImageIcon ii = new ImageIcon(image);            //得到 ImageIcon 对象
13       private JLabel jlRoot=new JLabel(ii);           //创建 Jlabel 对象
14      private Manager mg;                              //登录管理员名
15      String mgNo;                                     //管理员 ID
16      CardLayout cl=new CardLayout();                  //创建卡片布局对象
17      public Root(String mgNo)
18      {                                                //构造器
19          this.mgNo=mgNo;                              //获得管理员 ID
20          mg=new Manager(mgNo);                        //创建管理员管理面板
21          this.setManager();                           //设置管理员权限
22          this.initJp();                               //初始化卡片布局面板
23          this.addTreeListener();                      //为树节点注册事件监听器
24          for(int i=1;i<9;i++){
25              dtm.insertNodeInto(dtmn[i],dtmn[0],i-1); //向根节点添加子节点
26          }
27          jt.setEditable(false);           //设置该树中节点是可编辑的
28          this.add(jsplr);                 //将包含树的滚动窗口添加进窗体
29          jsplr.setLeftComponent(jt);      //将滚动窗口添加进左边的子窗口
30          jp.setBounds(200,50,600,500);    //为 jp 设置大小、位置，并添加进右边的子窗口
31          jsplr.setRightComponent(jp);     //将 jp 添加到分割窗体的右部
32          jsplr.setDividerLocation(200);   //设置分隔条的初始位置
33          jsplr.setDividerSize(4);         //设置分隔条的宽度
34          jlRoot.setFont(new Font("Courier",Font.PLAIN,30));//设置标签的字体
35           jlRoot.setHorizontalAlignment(JLabel.CENTER);     //设置水平对齐方式
36           jlRoot.setVerticalAlignment(JLabel.CENTER);       //设置竖直对齐方式
37           this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //设置窗体关闭动作
38           Image image=new ImageIcon(this.getClass().getResource("/ico.png")).
             getImage();                                       //加载图标对象
39           this.setIconImage(image);                         //设置窗体图标
40           this.setTitle("智达图书馆管理系统");               //设置窗体标题名称
41           Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
                                                               //得到屏幕尺寸
42           int centerX=screenSize.width/2;                   //得到水平方向中心
43           int centerY=screenSize.height/2;                  //得到竖直方向中心
44           int w=500;                                        //本窗体宽度
45           int h=400;                                        //本窗体高度
46           this.setBounds(centerX-w/2,centerY-h/2-100,w,h);//设置窗体出现在屏幕中央
47            this.setExtendedState(JFrame.MAXIMIZED_BOTH);    //窗体全屏
48            this.setVisible(true);                  //设置窗体可见
49            jt.setShowsRootHandles(true);           //设置显示根节点的控制图标
50      }
51      public static void main(String args[]){new Root("wyf");}  //main 方法
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
52  }
53  class NodeValue{
54      String value;                                   //自定义节点对象字符属性
55      public NodeValue(String value) {                //构造器
56          this.value=value;                           //得到节点的值
57      }
58      public String getValue(){                       //value 的 Get 方法
59          return this.value;                          //返回节点的值
60      }
61      public String toString(){                       //重写 toString 方法
62          return value;                               //返回值 value
63  }}
```

- 第 2～16 行声明和创建了搭建管理端界面用到的一些控件和使用的一些其他模块类的对象。
- 第 17～50 为管理端界面的构造器，主要是初始化界面，摆放控件的一些位置，第 49 行是用于测试的 main 方法，实际使用进入本界面是从登录界面登录而来的，这里的 main 方法只是为了测试方便而开发的。
- 第 53～63 行为自定义的一个类，用于在创建 DefaultMutableTreeNode 对象时对该节点对象进行初始化，在后面的监听事件中根据 MyNode 对象的属性值来判断单击哪一个节点，从而确定显示哪一个功能模块。

提示：在上述代码的构造器里，有些功能是通过调用方法实现的，这些方法将在下面介绍。

28.6.2 管理端其他方法介绍

28.6.1 节主要介绍了管理端主窗体界面搭建所用的一些控件和卡片布局面板初始化等的开发，本节将介绍其构造器中调用的一些方法，具体代码如下。

代码位置：见光盘中本章源代码的 StudentSystem.java 类中的 StudentSystem 方法。

```
1   public void setManager(){
2       String sql="select permitted from manager where mgNo='"+mgNo+"'";
                                                        //搜索管理员权限的 SQL
3       DataBase db=new DataBase();                     //创建数据库类对象
4       db.selectDb(sql);                               //执行查询
5       try{
6           db.rs.next();                               //结果集游标下移
7           String str=db.rs.getString(1).trim();       //得到管理员权限
8           if(str.equals("0")){mg.setFlag(false); }    //设置管理员权限
9       }
10      catch(Exception e){e.printStackTrace();}        //捕获异常并打印
11  }
12  public void initJp(){                               //初始化 jp
13      jp.setLayout(cl);                               //设置布局管理器为卡片布局
14      jp.add(jlRoot,"root");                          //添加根节点显示信息
15      jp.add(new Student(),"stu");                    //添加学生管理模块界面
16      jp.add(new BookManage(),"bm");                  //添加图书管理模块界面
17      jp.add(new SearchBook(),"sb");                  //添加查找图书管理界面
18      jp.add(new BorrowBook(),"bb");                  //添加借阅预约图书模块界面
19      jp.add(new ReturnBook(),"rb");                  //添加归还挂失图书界面
20      jp.add(this.mg,"Manager");                      //添加管理员管理模块界面
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



```
21        jp.add(new ExceedTime(),"et");                    //添加罚款处理界面
22    }
23    public void addTreeListener(){
24        jt.addTreeSelectionListener(new TreeSelectionListener(){
                                                            //为树添加选择模型监听器
25                public void valueChanged(TreeSelectionEvent e){ //当值变化时
26                    DefaultMutableTreeNode cdmtn=         //得到选中的节点对象
27                                (DefaultMutableTreeNode)e.getPath().getLast-
                                PathComponent();
28                    NodeValue cnv=(NodeValue)cdmtn.getUserObject();
                                                            //得到自定义节点对象
29                    if(cnv.value.equals("图书馆管理系统")){
                                                            //单击“图书馆管理系统”节点
30                        cl.show(jp,"root");               //显示根节点信息
31                    }
32                 if(cnv.value.equals("学生用户管理")){     //单击“学生用户管理”节点
33                        cl.show(jp,"stu");                //显示学生用户管理界面
34                    }
35                    else if(cnv.value.equals("图书管理")){   //单击“图书管理”节点
36                       cl.show(jp,"bm");                  //显示图书管理界面
37                    }
38                    if(cnv.value.equals("查询图书"))      //单击“查询图书”节点
39                        cl.show(jp,"sb");                 //显示查询图书界面
40                    }
41                    else if(cnv.value.equals("借阅预约图书")){
                                                            //单击“借阅预约图书”节点
42                        cl.show(jp,"bb");                 //显示借阅预约图书界面
43                    }
44                    else if(cnv.value.equals("归还挂失图书")){
                                                            //单击“归还挂失图书”节点
45                        cl.show(jp,"rb");                 //显示归还挂失图书界面
46                    }
47                    else if(cnv.value.equals("交纳罚款"))  {//单击“交纳罚款”节点
48                        cl.show(jp,"et");                 //显示缴纳罚款界面
49                    }
50                    else if(cnv.value.equals("管理员管理")){ //单击“管理员管理”节点
51                        cl.show(jp,"Manager");            //显示管理员管理界面
52                    }
53                    else if(cnv.value.equals("退出")){  //显示退出界面
54                        int i=JOptionPane.showConfirmDialog(Root.this,"是否退出
55                        系统?", "消息",JOptionPane.YES_NO_OPTION);
                          //弹出对话框
56                        if(i==JOptionPane.YES_OPTION){System.exit(0);}
                                                            //退出系统
57        }}}});}
```

- 第 1～11 行为从数据库得到该管理员的权限，并根据权限将管理员管理模块界面的一些控件禁用，使没有权限的管理员不能添加和删除管理员。
- 第 12～22 行初始化了卡片布局面板，主要是将一些功能模块的界面添加进此面板。
- 第 23～57 行给树节点注册了事件监听器，并编写监听代码，实现了选中树上不同的节点，右侧卡片布局面板根据选中的节点显示相应的功能界面。

至此，主界面的搭建工作已经完成，由于各个功能模块还没有进行开发，所以还不能编译进行测试，待后续章节将各个模块开发完毕后，再进行集成测试。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

28.7 图书查询模块

28.6 节主要介绍了管理系统主界面框架的搭建，本节及后续的内容将分别介绍各功能模块的开发。下面首先对查询图书功能模块的开发进行介绍，查询图书界面效果如图 28-17 所示。

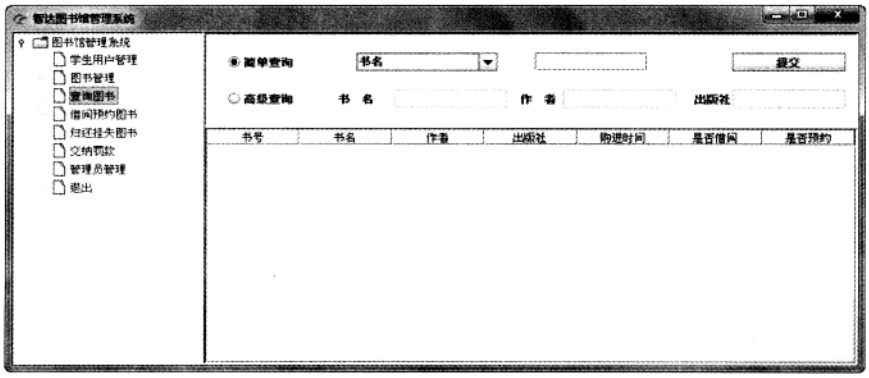


图 28-17 查询图书

28.7.1 查询界面

当学生用户在学生信息系统中单击“查询图书”节点时，就会进入查询图书页面，其效果图见 28-4，页面开发的详细代码如下。

代码位置：见光盘中本章源代码的 SearchBook.java 类。

```
1   public class SearchBook extends JFrame implements ActionListener{
2       int flag;                                          //声明标志位
3       String sql;                                        //声明 SQL 字符串引用
4       DataBase db;                                       //声明 DataBase 对象引用
5       private JSplitPane jsp=new JSplitPane(JSplitPane.VERTICAL_SPLIT,true);
                                                           //分割方向为上下
6       private JPanel jpt=new JPanel();                   //创建窗体上部 JPanel 对象
7       private JPanel jpb=new JPanel();                   //创建窗体下部 JPanel 对象
8       private String[] str={"书名","出版社","作者","购买时间"};
                                                           //创建下拉列表框数据的字符串数组
9       private JComboBox jcb=new JComboBox(str);          //创建下拉列表框
10      private JButton jb=new JButton("提交");            //创建按钮
11      private JLabel[] jlArray=new JLabel[]{             //创建并初始化标签数组
12          new JLabel("  书    名"),new JLabel("  作    者"),new JLabel("出版社")};
13      private JTextField[] jtxtArray=new JTextField[]{//创建并初始化文本框数组
14          newJTextField(),newJTextField(),newJTextField(),newJTextField()};
15      private JRadioButton[] jrbArray={                  //创建并初始楷单选按钮数组
16          new JRadioButton("简单查询",true),new JRadioButton("高级查询")};
17      private ButtonGroup bg=new ButtonGroup();          //创建按钮组
18      Vector<String> head = new Vector<String>();        //创建表头向量
19      {head.add("书号");head.add("书名");head.add("作者");head.add("出版社");
20      head.add("购进时间");head.add("是否借阅");head.add("是否预约");}//初始化表头
21      Vector<Vector> data=new Vector<Vector>();          //存放检索出的书的基本信息
22      DefaultTableModel dtm=new DefaultTableModel(data,head);  //创建表格模型
23      JTable jt=new JTable(dtm);                         //创建 Jtable 对象
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
24        JScrollPane jspn=new JScrollPane(jt);          //将 Jtable 添加到滚动窗格
25       public void actionPerformed(ActionEvent e){} //实现 actionPerformed 方法
26    }
```



说明：上面的代码主要是为“查询图书”页面的搭建创建标签、按钮、下拉列表框、文本框、单选按钮等控件，同时在分割窗体的下部创建表格模型，以便查询成功后将查询结果显示在表格里。

28.7.2　查询界面构造器

28.7.1 节主要是对“查询图书”界面的搭建创建了一些控件和表格模型，本节将介绍其构造器的开发，具体代码如下。

代码位置：见光盘中本章源代码的 SearchBook.java。

```
1   public SearchBook(){
2           this.setLayout(new GridLayout(1,1));    //设置查询图书界面为网格布局 3
3           jpt.setLayout(null);  //设置整个 RetrunBook 界面上部分均为空布局管理器
4           jpb.setLayout(null);  //设置整个 RetrunBook 界面下部分均为空布局管理器
5           jpt.add(jcb);                                   //将其添加到面板中
6           jcb.setBounds(160,20,150,20);                   //设置大小及位置
7          jcb.addActionListener(this);                     //注册事件监听器
8           jpt.add(jb);                                    //将按钮添加到面板中
9           jb.setBounds(560,20,120,20);                    //设置面板的大小及位置
10          jb.addActionListener(this);                     //为按钮注册事件监听器
11          for(int i=0;i<2;i++){                           //对单选按钮进行设置
12              jrbArray[i].setBounds(20,20+i*40,100,20);//设置单选按钮的大小及位置
13              jpt.add(jrbArray[i]);                       //将单选按钮添加到面板中
14              jrbArray[i].addActionListener(this);    //为单选按钮注册事件监听器
15              bg.add(jrbArray[i]);                        //将单选按钮添加到按钮组里面
16          }
17          for(int i=0;i<3;i++){                           //循环标签数组
18              jlArray[i].setBounds(120+i*200,60,80,20);//设置标签的大小及位置
19              jtxtArray[i].setBounds(200+i*180,60,120,20);
                                                            //设置文本框的大小及位置
20              jpt.add(jtxtArray[i]);                      //将标签添加到上部面板中
21              jpt.add(jlArray[i]);                        //将文本框添加到面板中
22          }
23          for(int i=0;i<3;i++){                           //遍历文本框数组
24              jtxtArray[i].setEditable(false);    //设置文本框为不可用
25          }
26          jtxtArray[3].setBounds(350,20,120,20); //设置文本框的大小及位置
27          jpt.add(jtxtArray[3]);                          //将文本框添加到面板中
28        jsp.setTopComponent(jpt);                         //把 jpt 设置到 jsp 的上部窗格
29        jsp.setBottomComponent(jspn);                     //设置分割窗体下部为 jspn
30        jsp.setDividerSize(4);                            //设置分割条的大小
31          this.add(jsp);                                  //将分割窗体添加到窗口中
32         jsp.setDividerLocation(100);                     //设置 jsp 中分割条的初始位置
33        this.setBounds(3,10,600,400);                     //设置窗体的大小及位置
34        this.setVisible(true);                            //设置窗体的可见性
35      }
```

- 第 2 行代码是设置本面板为网格布局，这样就可以用该面板把卡片布局的面板占满，以提高视觉效果。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

- 第 6～10 行、第 11～16 行、第 17～22 行分别设置了单选框、按钮、单选按钮的大小和位置，并为其注册了事件监听器。
- 第 23～25 行代码是设置 jtxtArray[0]～jtxtArray[2]为不可编辑状态，这是为了让用户在进行“简单查询”时，避免将查询信息输入到“高级查询”的输入文本框中。当用户需要进行“高级查询”时，这几个文本框将被设为可编辑状态。

提示：后面其他功能模块的构造器与本节构造器的结构和功能类似，因此，在后面的功能模块中，构造器的讲解会一带而过，讲解重点将放在事件响应等核心代码上。

28.7.3 简单查询

28.7.2 节主要介绍了 SearchBook 构造器的开发过程，本节将重点介绍 “简单查询”功能的开发过程，其详细代码如下所列。

提示：当进入“简单查询”功能模块后，学生用户可以根据书名、出版社、作者或购进时间之中任一条件对图书信息进行查询。查询成功后，将查询结果显示在页面下部的表格中。

代码位置：见光盘中本章源代码的 SearchBook.java 类中的 actionPerformed 方法。

```
1    if(jrbArray[0].isSelected()){                    //选中“简单查询”单选按钮
2    jtxtArray[3].setEditable(true);                  //将 jtxtArray[3]设置为可编辑状态
3    for(int i=0;i<jtxtArray.length-1;i++){
4        jtxtArray[i].setEditable(false);             //设置文本框为不可编辑
5    }
6    if(jcb.getSelectedIndex()>=0&&jcb.getSelectedIndex()<4){
                                                      //当选择索引在 0~3 之间时
7        jtxtArray[3].requestFocus();                 //输入文本框得到输入焦点
8      if(e.getSource()==jb){                         //如果事件源为“提交”按钮，则执行检索
9           String str=jtxtArray[3].getText().trim();   //得到查询条件
10          if(str.equals("")){                        //查询条件为空
11              JOptionPane.showMessageDialog(this,"请输入必要的信息!!! ",
12                                  "消息",JOptionPane.INFORMATION_MESSAGE);
                                                      //弹出信息提示框
13              return;                                //返回
14          }
15         if(jcb.getSelectedIndex()==0){              //根据书名进行查询
16              sql="select * from BOOK where BookName='"+str+"'";
                                                      //检索书名为 str 的图书信息
17          jtxtArray[3].setText("");                  //将 jtxtArray[3]的内容设置为空
18          }
19          else if(jcb.getSelectedIndex()==1){//根据出版社进行查询
20              sql="select * from BOOK where Publishment='"+str+"'";
                                                      //检索出版社为 str 的图书信息
21              jtxtArray[3].setText("");              //将 jtxtArray[3]的内容设置为空
22          }
23          else if(jcb.getSelectedIndex()==2){         //根据作者进行查询
24              sql="select * from BOOK where Author='"+str+"'";
                                                      //检索作者为 str 的书的信息
25              jtxtArray[3].setText("");              //将 jtxtArray[3]的内容设置为空
26          }
27          else{                                      //根据购进时间进行查询
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



```
28              sql="select * from BOOK where BuyTime='"+str+"'";
                                                    //检索购买时间为 str 的图书信息
29              jtxtArray[3].setText("");           //将 jtxtArray[3]的内容设置为空
30          }
31          db=new DataBase();                      //创建数据库类对象
32          try{
33              sql = new String(sql.getBytes(),"ISO-8859-1");  //将 SQL 转码
34          }
35          catch(Exception ae){ae.printStackTrace();}          //捕获异常并打印
36          db.selectDb(sql);                                   //执行查询
37         }}
38      Vector<Vector> vtemp = new Vector<Vector>();//创建向量,用于存放表格数据
39      try{
40          int flag=0;                             //判断是否查询到内容
41          while(db.rs.next()){                    //取到结果集
42              flag++;                             //标志位自加
43              Vector<String> v = new Vector<String>();
                                                    //创建向量,用于存放每一行的数据
44              for(int i=1;i<=7;i++){              //将每列添加到临时数组 v
45                  String str1=db.rs.getString(i);
46                  str1=new String(str.getBytes("ISO-8859-1"),"gb2312");
                                                    //将 ISO-8859-1 转为 gb2312
47                  v.add(str1);                    //添加到表格数据向量中
48              }
49              vtemp.add(v);                       //将各条记录添加到 vtemp
50          }
51          if(flag==0){                            //没有查询到内容
52              JOptionPane.showMessageDialog(this,"没有您要查找的内容!!! ",
53                      "消息",JOptionPane.INFORMATION_MESSAGE);
                                                    //弹出信息提示对话框
54             return;                              //返回
55          }}
56      catch(Exception ea){ea.printStackTrace();}  //捕获异常并打印
57      dtm.setDataVector(vtemp,head);              //更新 table
58      jt.updateUI();                              //提示表格已经更改
59      jt.repaint();                               //重绘表格
60      db.dbClose();                               //关闭数据库连接
61  }
```

- 第 1～37 行代码主要完成对图书信息查询功能的开发。第 1 行代码判断“简单查询”单选按钮是否被选中；第 8 行指出如果事件源为“提交”按钮，则执行检索；第 10～14 行是对输入为空的情况进行处理。
- 第 15～30 行分别根据四种查询条件之一进行查询；第 33 行是对查询的 SQL 语句进行转码。
- 第 39～61 行代码完成了把查询结果显示在表格中的开发。第 46 行是对结果集中的数据进行转码，第 51～55 行代码对查询内容为空进行了处理，并弹出消息对话框。



提示：第 33 行和第 46 行代码进行了转码，使用 MySQL 数据库时，如果不进行转码，当向数据库插入或从数据库得到中文字符时，就有可能出现乱码。

28.7.4　高级查询

前面实现了查询图书界面的搭建及简单查询功能的开发，本节将对高级查询功能进行具体的开发，其代码如下所列。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

代码位置：见光盘中本章源代码的 SearchBook.java 类。

```
1   if(jrbArray[1].isSelected()){                    //“高级查询”单选按钮被选中
2       jtxtArray[0].requestFocus();                 //获得输入焦点
3       jtxtArray[3].setEditable(false);             //将简单查询所涉及的文本框设为可编辑
4       for(int i=0;i<jtxtArray.length-1;i++){ //将高级查询所涉及的文本框设为可编辑
5           jtxtArray[i].setEditable(true);//将文本框设置为可编辑状态
6           }
7       if(e.getSource()==jb){                                    //单击“提交”按钮
8          int bz=this.seniorSearch();                            //得到标志位
9          if(bz!=0){return;}                                     //查询条件不对
10         db=new DataBase();                                     //创建数据库对象
11         db.selectDb(sql);                                      //执行查询
12         Vector<Vector> vtemp = new Vector<Vector>();           //创建向量集合
13         try{
14             int flag=0;                                        //结果集记录的条数
15             while(db.rs.next()){                               //取到结果集
16                 flag++;                                        //标志位自加
17                 Vector<String> v = new Vector<String>();    //创建临时向量
18                 for(int i=1;i<=7;i++){     //将每列添加到临时向量集合 v
19                     String str=db.rs.getString(i);     //得到第 i 列数据
20                     str=new  String(str.getBytes("ISO-8859-1"),"gb2312");
                                                          //转码
21                     v.add(str);                        //将信息添加到向量
22                 }
23                 vtemp.add(v);              //将各条记录添加到临时向量 vtemp
24             }
25             if(flag==0){                   //查询失败，则弹出提示对话框
26                 JOptionPane.showMessageDialog(this,"没有您要查找的内容!!! ",
27                             "消息",JOptionPane.INFORMATION_MESSAGE);
                                              //弹出提示框
28                return;                     //返回
29             }}
30         catch(Exception ea){ea.printStackTrace();}      //捕获异常并打印
31         dtm.setDataVector(vtemp,head);                  //设置表格数据
32         jt.updateUI();jt.repaint();                     //更新 table
33         db.dbClose();                                   //关闭数据库连接
34      }}
```

- 第 1～34 行代码实现了“高级查询”功能的开发。第 1 行代码是指选中“高级查询”单选按钮；第 3 行代码将简单查询所涉及的文本框设为不可编辑状态；第 4～6 行是将高级查询所涉及的文本框设为可编辑状态。
- 单击“提交”按钮将执行查询功能，在查询过程中调用了 seniorSearch 方法。第 7～34 行代码实现了当查询成功后将查询结果显示在表格中，查询失败则给出提示的功能。

上述代码中用到了 seniorSearch 方法，该方法首先判断查询条件是否为空，然后将用户输入的查询条件拼装成 SQL 语句。该方法返回一个整型变量来表示用户输入查询条件是否为空。具体代码如下。

代码位置：见光盘中本章源代码的 SearchBook.java 类。

```
1   public int seniorSearch(){
2       int flag=0;                                    //设置标志位
3       String str0=jtxtArray[0].getText().trim();  //得到用户输入的书名查询条件
4       String str1=jtxtArray[1].getText().trim();  //得到用户输入的作者查询条件
5       String str2=jtxtArray[2].getText().trim();  //得到用户输入的出版社查询条件
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
6        if(str0.equals("")&&str1.equals("")&&str2.equals("")){   //文本框输入为空
7            JOptionPane.showMessageDialog(this,"请输入必要的信息!!! ",
8                                    "消息",JOptionPane.INFORMATION_MESSAGE);
                                                        //弹出对话框
9            flag++;                                    //标志位自加
10       }
11       if(((!str0.equals(""))&&(str1.equals(""))&&(str2.equals("")))
12           ||((str0.equals(""))&&(!str1.equals(""))&&(str2.equals("")))
13           ||((str0.equals(""))&&(str1.equals(""))&&(!str2.equals("")))){
14           JOptionPane.showMessageDialog(this,"请使用简单查询!!! ",
15                                   "消息",JOptionPane.INFORMATION_MESSAGE);
                                                        //弹出对话框
16           flag++;                                    //标志位自加
17       }
18      if((!str0.equals(""))&&(!str1.equals(""))&&(str2.equals(""))){
                                                        //书名和作者组合
19           sql="select * from BOOK where BookName='"+str0+"' and
             Author='"+str1+"'";                        //组合成 SQL 语句
20           jtxtArray[0].setText("");jtxtArray[1].setText("");   //清空文本框
21       }
22       if((!str0.equals(""))&&(str1.equals(""))&&(!str2.equals(""))){
                                                        //书名和出版社
23           sql="select * from Book where BookName='"+str0+"' and
             Publishment='"+str2+"'";                   //组合成 SQL
24           jtxtArray[0].setText("");jtxtArray[2].setText("");   //清空文本框
25       }
26       if((str0.equals(""))&&(!str1.equals(""))&&(!str2.equals(""))){
                                                        //作者与出版社
27           sql="select * from Book where Author='"+str1+"' and Publishment=
             '"+str2+"'";                               //组合成 SQL
28           jtxtArray[1].setText("");jtxtArray[2].setText("");   //清空文本框
29       }
30       if((!str0.equals(""))&&(!str1.equals(""))&&(!str2.equals(""))){//三者组合
31           sql="select * from Book where BookName='"+str0
32                       +"' and Publishment='"+str2+"' and Author='"+str1+"'";
                                                        //组合成 SQL 语句
33           jtxtArray[0].setText("");jtxtArray[1].setText("");jtxtArray[2].
             setText("");                               //将文本框清空
34       }
35       return flag;                                   //返回标志位
36   }
```

- 第 3～5 行是获得用户输入的一些查询条件，如作者、书名和出版社。
- 第 6～10 行判断查询条件是否为空，若为空，则弹出提示信息对话框，第 11～17 行判断用户是否只输入了一个查询条件，这与简单查询功能一样，提示用户使用简单查询功能。
- 第 18～34 根据三种查询条件的不同组合，拼装了不同的 SQL 语句，待查询数据库时调用。

至此，查询图书模块的开发已经基本完成，可以运行 SearchBook 类中的 main()方法进行测试。



28.8　图书借阅预约模块

28.7 节主要介绍了“查询图书”功能的开发，本节将介绍本系统中另一个重要的模块——

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

“借阅预约图书”功能模块，开发完后的界面如图 28-18 所示。



图 28-18　借阅预约图书

28.8.1　界面开发

同“查询图书”模块类似，本模块首先为读者介绍界面的开发，具体是介绍如何使用 Swing 中的标签、文本框、按钮等控件，并对其进行组合搭配的过程，其开发代码如下。

代码位置：见光盘中本章源代码的 BorrowBook.java 类。

```
1   public class BorrowBook extends Jframe implements ActionListener{
2      ……//此处省略了界面的搭建代码
3       public void actionPerformed(ActionEvent e){}      //动作事件处理方法
4   }
```

说明：这里与上一节界面的搭建类似，由于篇幅所限，对该界面的开发代码就不再详细讲解。读者请参阅光盘中本章源代码的 BorrowBook.java。

28.8.2　显示图书信息

28.8.1 节简单介绍了“借阅预约图书”界面的开发，本节主要介绍显示图书信息功能的开发，其详细代码如下。

代码位置：见光盘中本章源代码的 BorrowBook.java 类。

```
1   if(e.getSource()==jb2){
2       if(jtxt4.getText().equals(""))                    //对输入为空的情况进行处理{
3           JOptionPane.showMessageDialog(this,"输入不能为空，请重新输入!!! ",
4                                   "信息",JOptionPane.INFORMATION_MESSAGE);//弹出对话框
5           return;                                       //返回
6       }
7       //查询学号文本中所输学号是否存在于 Student 表中
8       sql="select * from STUDENT where StuNO="+Integer.parseInt(jtxt4.getText().
        trim());
9       db=new DataBase();                                //创建数据库操作类对象
10      db.selectDb(sql);                                 //执行查询
11      Vector<Vector> vtemp = new Vector<Vector>();//创建向量,用于存放表格数据
12      try{
13          if(!(db.rs.next())){                          //若学号错误，输出提示对话框
14              JOptionPane.showMessageDialog(this,"输入了错误的学号","消息",
15                          JOptionPane.INFORMATION_MESSAGE);    //弹出对话框
16          }
17          else{                                         //得到此学号学生的姓名和班级
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



```
18              String stuName=db.rs.getString(2).trim();   //得到学生姓名
19              String classes=db.rs.getString(5).trim();   //得到学生班级
20              stuName = new String(stuName.getBytes("ISO-8859-1"),"gb2312");
                                                            //将编码转为 gb2312
21              classes = new String(classes.getBytes("ISO-8859-1"),"gb2312");
                                                            //将编码转为 gb2312
22              //若学号正确，则检查该学生是否有权限借书或预约
23              if(db.rs.getString(8).trim().equals("否")){
                                                        //若无权限则输出提示对话框
24                  JOptionPane.showMessageDialog(this,"您无此权限!! ","消息",
25                          JOptionPane.INFORMATION_MESSAGE);//弹出对话框
26              }
27              else{                                       //若有权限
28                  db.selectDb(sql);                       //执行查询
29                  do{
30                      String str6=null;     //str6 存入 Book 表记录中的第 6 项
31                      String str7=null;     //str7 存入 Book 表记录中的第 7 项
32                      String bookName=null; //定义输入书号所对应书的书名
33                      String author=null;   //定义输入书号所对应书的作者
34                      if(!(db.rs.next())){  //没有该书号，则输出提示对话框
35                          JOptionPane.showMessageDialog(this,"没有您要查
36                          找的内容","消息",JOptionPane.INFORMATION_MESSAGE);
                            //弹出对话框
37                      }
38                      Vector<String> v = new Vector<String>();     //创建向量
39                      for(int i=1;i<=7;i++)
40                      {//顺序得到所搜到的结果中的各项记录
41                          if(i==5){          //把结果集的第六条信息添加进临时向量
42                              str6=db.rs.getString(i+1);//得到结果集中第 6 项
43                              str6=new String(str6.getBytes("ISO-8859-1"),
44                              "gb2312");v.add(str6);
                                //转为 gb2312，添加到向量中
45                          }
46                          if(i==6){          //把结果集的第七条信息添加进临时向量
47                              str7=db.rs.getString(i+1);//得到结果集中的第 7 项
48                              str7=new String(str7.getBytes("ISO-8859-1"),
49                              "gb2312");v.add(str7);     //转码，添加到向量中
50                          }
51                          if(i==2){          //把结果集的第二条信息添加进临时向量
52                              bookName=db.rs.getString(i).trim();//得到书名
53                      bookName=new  String(bookName.getBytes("ISO-8859-1"),
54                      "gb2312"); v.add(bookName);         //转码，添加到向量中
55                          }
56                          if(i==3){          //把结果集的第三条信息添加进临时向量
57                              author=db.rs.getString(i).trim();  //得到作者
58                          author=new  String(author.getBytes("ISO-8859-1"),
59                          "gb2312");v.add(author);        //转码，添加到向量中
60                          }
61                          if(i==1){          //把结果集的第一条信息添加进临时向量
62                              String str=db.rs.getString(i).trim();
                                                //得到第 1 条信息
63                              str=new String(str.getBytes("ISO-8859-1"),
                                "gb2312");    //转码
64                              v.add(str);   //添加到向量中
65                          }
66                          if(i==4){          // 把结果集的第四条信息添加进临时向量
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
67                                  String str=db.rs.getString(i).trim();
                                                    //得到第 4 条信息
68                                  str=new String(str.getBytes("ISO-8859-1"),
69                                  "gb2312");v.add(str);      //转码，添加到向量中
70                              }}
71                          vtemp.add(v);           //更新结果框中的内容
72                          dtm.setDataVector(vtemp,head);      //设置表格显示数据
73                          jt.updateUI();jt.repaint();         //更新表格
```

- 第 1～6 行代码是对用户输入信息为空时进行提示，此时将会弹出一个提示框。
- 第 8～16 行用来判断该学生的学号是否正确，若错误则弹出信息提示对话框。
- 第 17～26 行从数据库得到该学生的基本信息，并判断该学生是否有借书的权限，如果没有，则提示权限不够。
- 第 34～37 行 SQL 语句用于查找所输入的书号是否存在于 Book 表中，若结果集为空，则给出提示。
- 第 38～73 行代码是把搜索结果转码后添加进向量，然后显示在页面下部的表格中。



说明：本节主要完成对借书或预约图书的人的身份及权限确认功能的开发，只有本校学生用户并且该生未欠款这两个条件都满足，才能够借阅或预约图书。同时本节还实现了将搜索出的图书信息显示在页面下部的表格中。

28.8.3 借阅预约功能

28.8.2 节主要实现了对借阅预约图书的人的身份及权限进行确认，将搜索出的图书信息进行显示功能的开发，本节将对“借阅预约图书”的功能进行开发，详细代码如下。

代码位置：见光盘中本章源代码的 BorrowBook.java 类。

```
1       if(jrbArray[0].isSelected()){       //选择了借图书
2           if(str6.trim().equals("是")){  //若用户想借的书已经被借走，输出提示信息框
3               JOptionPane.showMessageDialog(this,
4                   "此书已经被借","消息",JOptionPane.INFORMATION_MESSAGE);
                                            //弹出信息提示框
5           }
6           else if(str7.trim().equals("是")){ //用户想借的书已被预约不能借，给出提示
7               JOptionPane.showMessageDialog(this,
8                 "此书已经被预约,不能再借","消息",JOptionPane.INFORMATION_MESSAGE);
                                            //弹对话框
9           }
10          else{
11              Date now =new Date();       //获得当前日期来记录借书时间和应还时间
12              sql="update BOOK set Borrowed='是' where BookNO="
13                                  +Integer.parseInt(jtxt3.getText().trim());
                                      //更新 BOOK 表的 SQL 语句
14              db.updateDb(sql);       //设置该书的 Borrowed 项为“是”,弹出成功信息框
15                  JOptionPane.showMessageDialog(this, "借书成功",
16                          "消息",JOptionPane.INFORMATION_MESSAGE);//弹出对话框
17               sql="insert into RECORD values("+Integer.parseInt(jtxt3.
                 getText().trim())+","
18                  +Integer.parseInt(jtxt4.getText().trim())+",'"+(now.
                    getYear()+1900)+"."
19                  +(now.getMonth()+1)+"."+now.getDate()+"',"+"'"+(now.
                    getYear()+1900)+"."
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
20                  +(now.getMonth()+2)+"."+now.getDate()+"','否','否')";
21              db.updateDb(sql);      //将该书记录插入 Record 表中
22          }
23      }
24      if(jrbArray[1].isSelected()){                    //选择了预约图书
25          if(str7.trim().equals("是")){                //该书已经被预约，输出提示信息框
26              JOptionPane.showMessageDialog(this,
27                  "此书已经被预约","消息",JOptionPane.INFORMATION_MESSAGE);
                                                         //弹出对话框
28          }
29          else{                                        //预约成功设置预约项 Ordered
30              sql="update BOOK set Ordered='是' where BookNO="
31                  +Integer.parseInt(jtxt3.getText().trim());//更新 BOOK 表的 SQL
32              db.updateDb(sql);                        //执行更新操作
33              JOptionPane.showMessageDialog(this,
34                  "预约成功","消息",JOptionPane.INFORMATION_MESSAGE);
                                                         //弹出预约成功信息框
35              sql="insert into ORDERREPORT values("+Integer.parseInt(jtxt3.
                getText().trim())
36                  +",'"+stuName+"','"+classes+"','"+bookName+"',"
37                  +Integer.parseInt(jtxt4.getText().trim())+",'"+author+"')";
38              db.updateDb(sql);                        //执行更新
39          }}}while(db.rs.next());
40      }}}
41      catch(Exception ex){ex.printStackTrace();}  //捕获异常并打印
42      db.dbClose();                                //关闭数据库连接
43  }
```

- 第 1～23 行代码是对图书借阅功能的开发。第 2～9 行是对用户想借的书已被借走或被预约情况的处理。第 11～23 行是对借阅功能的具体开发，首先将 Book 表中该书是否被借阅标志设置为“是”，同时将借书信息添加进 Record 表，并弹出“借书成功”提示框。
- 第 24～40 行是对预约图书功能的开发。第 25～28 行是对该书已被预约情况的处理。第 29～40 行是对预约功能的具体开发，首先将 Book 表中该书的“Orderred”项设置为“是”，同时将预约信息添加进 OrderRecort 表中，并弹出 “预约成功”信息对话框。



提示：前面主要实现了判断该学生用户是否有借阅预约图书的权限，而本节是具体借阅预约功能的开发。本节与 28.8.2 节的代码其实是一个方法内的，只是将它拆开来进行讲解。

至此，图书借阅预约功能模块的开发已经基本完成，可以运行 BorrowBook 类中的 main() 方法进行测试。



28.9　图书归还挂失模块

28.8.3 节介绍了图书借阅预约功能模块，它实现了学生用户借阅和预约图书的功能，本节将详细介绍“归还挂失图书”功能模块的开发，图书归还挂失界面开发完的运行效果如图 28-19 所示。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

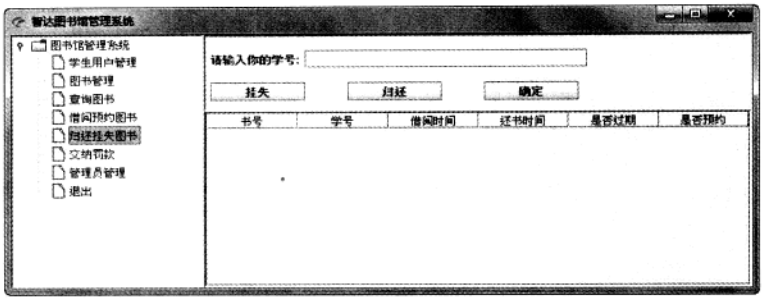


图 28-19　归还挂失界面

28.9.1　界面开发

开发“归还挂失图书”功能模块之前，首先进行界面的搭建，本节需要搭建的是归还挂失图书的界面，具体代码如下。

代码位置：见光盘中本章源代码的 ReturnBook.java 类。

```
1   public class ReturnBook extends JFrame implements ActionListener {
2       ……//此处省略了界面的搭建代码
3       public void actionPerformed(ActionEvent e){}     //事件处理方法
4   }
```

说明：本界面的开发与“借阅预约图书”界面的开发类似，因为篇幅所限，在这里就不再赘述。

28.9.2　查询借书记录

学生借阅了图书，可随时在本系统查询自己的借书记录，本节将介绍查询借书记录功能的开发，具体代码如下。

代码位置：见光盘中本章源代码的 ReturnBook.java 类中的 actionPerformed 方法。

```
1   if(e.getSource()==jbArray[2]){                                    //事件源为“确定”按钮
2       if(jtxt.getText().trim().equals("")){                         //学号输入为空，则提示
3           JOptionPane.showMessageDialog(this,"请输入学号",
4                       "消息",JOptionPane.INFORMATION_MESSAGE);//弹出对话框
5               return;                                               //返回
6       }
7       else{                                                         //根据学号进行查询
8           sql="select * from RECORD where StuNO="+jtxt.getText().trim();
                                                                      //查询某学生的记录
9           db=new DataBase();                                        //创建数据库对象
10          db.selectDb(sql);                                         //从数据库搜索
11          Vector<Vector> vtemp = new Vector<Vector>();     //创建向量
12          try{                                                  //结果集进行异常处理
13           int k=0;                                             //用来标志是否得到结果
14              while(db.rs.next()){                              //遍历结果集
15                 k++;                                           //标志位自加
16                  Vector<String> v = new Vector<String>();    //创建向量
17                  for(int i=1;i<7;i++){                         //将每列添加到临时向量 v
18                      String str=db.rs.getString(i);            //得到第 i 条记录
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
19                          str=new  String(str.getBytes("ISO-8859-1"),"gb2312");
                                                          //转码
20                          v.add(str);                   //添加到向量中
21                      }
22                      vtemp.add(v);                     //将各条记录添加到临时向量 vtemp
23                      jt.clearSelection();              //清空所选行
24                      dtm.setDataVector(vtemp,head);    //设置表格数据
25                      jt.updateUI();jt.repaint();       //更新 table
26                  }
27                  if(k==0){                             //弹出提示信息对话框
28                  JOptionPane.showMessageDialog(this,"输入了错误的学号或该学生没有借书
29                  记录","消息",JOptionPane.INFORMATION_MESSAGE);      //弹出对话框
30                      return;                           //返回
31                  }}
32              catch(Exception ea){ea.printStackTrace();} //捕获异常并打印
33          }}
```

- 第 2～6 行是对输入学号为空这种情况的处理。当输入不为空，将进行查询。第 7 行中的 SQL 语句就是根据学号在 record 表中进行搜索，以得到该生的借阅信息。
- 第 12～26 行代码是把搜索出的结果显示在页面下部的表格中。第 28～31 行代码用于处理结果集为空的情况，即该生没有借阅图书，此时将弹出提示对话框。

28.9.3　归还和挂失图书

28.9.2 节介绍了对用户借阅图书记录进行查询功能的开发，本节将介绍借阅记录查询出来以后，对借阅图书信息处理方法的开发，主要是归还图书和挂失图书，其开发代码如下。

代码位置：见光盘中本章源代码的 ReturnBook.java 类。

```
1   if(e.getSource()==jbArray[1]) {          //当要归还图书
2       int row=jt.getSelectedRow();         //得到选中的记录
3       if(row<0){                           //如果未选中下部表中的某些内容，进行提示
4           JOptionPane.showMessageDialog(this,"请选择要归还的书!!!","消息",
5                                   JOptionPane.INFORMATION_MESSAGE);//弹出对话框
6           return;                                                  //返回
7       }
8       str=(String)jt.getValueAt(row,0);                            //得到书号
9       int sno=Integer.parseInt((String)jt.getValueAt(row,1));  //得到学生学号
10      int bno=Integer.parseInt(str);                   //将书号转为 int 型
11      int flag=checkTime(sno,bno);                     //判断是否超期
12      if(flag==-1){                                    //如果图书超期
13          db=new DataBase();                           //创建数据库对象
14          sql="update student set permitted='否' where StuNO="+sno;
                                                         //更新学生借书权限的 SQL
15          db.updateDb(sql);                            //执行更新
16          db.dbClose();                                //关闭数据库连接
17      }
18      if(flag==0){return;}                             //如果图书未超期
19      sql="Delete from RECORD where BookNO="+Integer.parseInt(str);
                                                         //删除记录的 SQL
20      db=new DataBase();                               //创建 DataBase 对象
21      db.updateDb(sql);                                //执行更新
22      sql="update book set borrowed='否' where BookNO="+Integer.parseInt(str);
                                                         //更新书是否可借的 SQL
23      db.updateDb(sql);                                //更新了图书记录
24      db.dbClose();                                    //关闭连接
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
25        updateTable();                                          //更新表格
26    }
27    if(e.getSource()==jbArray[0]){                              //需要挂失图书
28        int row=jt.getSelectedRow();                            //得到所选择的行
29        if(row<0){                                              //当没有选中行时
30            JOptionPane.showMessageDialog(this,"请选择要挂失的书!!!","消息",
31                                    JOptionPane.INFORMATION_MESSAGE);
                                                                  //弹出提示信息
32            return;                                             //返回
33        }
34        loseBook(row);                                          //调用挂失书方法
35        updateTable();                                          //调用更新表格方法
36    }
```

说明：本节对借阅图书所做的处理主要有“归还”和“挂失”两种。当用户单击“归还”按钮时，将执行上面第 1～27 行代码；当单击“挂失”按钮时，将执行第 28～37 行代码。本节要注意三个 SQL 语句的使用，它们分别实现对学生信息的更新、对借阅图书记录的删除及对图书信息的更新。

28.9.4 挂失处理的开发

在 28.9.3 节中实现了对图书的归还和挂失功能的开发，本节将对所挂失图书的处理方法进行详细介绍，具体代码如下。

代码位置：见光盘中本章源代码的 ReturnBook.java 类中的 loseBook 方法。

```
1    public void loseBook(int row){
2            String bname="";                                    //用于记录书的名字
3            int lbno=0;
4            int bno=Integer.parseInt((String)jt.getValueAt(row,0));
                                                                 //得到丢失书的书号
5            String sno=(String)jt.getValueAt(row,1);            //得到丢失书的学生的学号
6            sql="select BookName from BOOK where BookNO="+bno;
                                                                 //搜索书名字的 SQL 语句
7            db=new DataBase();                                  //创建 DataBase 对象
8            db.selectDb(sql);                                   //执行检索
9            try{
10               if(db.rs.next()){                               //遍历结果集
11                   bname=db.rs.getString(1).trim();            //得到丢失书的书名
12               }}
13           catch(Exception e){e.printStackTrace();}            //打印异常
14           sql="select MAX(LbNO) from LoseBook";   //找到最大的丢失记录号的SQL语句
15           db.selectDb(sql);                                   //执行检索
16           try{
17               if(db.rs.next()){                               //遍历结果集中的记录
18                   lbno=db.rs.getInt(1);                       //得到最大的丢失记录号
19                   lbno++;                                     //自加
20               }}
21           catch(Exception ea){ea.printStackTrace();}  //捕获异常并打印
22       sql="insert into LOSEBOOK values("+lbno+","+sno+","+bno+",'"+bname+"')";
                                                                 //创建 SQL 语句
23           db.updateDb(sql);                                   //执行更新
24       sql="select BookNo from ORDERREPORT where BookNO="+bno;  //创建 SQL 语句
25           db.selectDb(sql);                                   //执行检索
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
26          try{
27              while(db.rs.next()){                          //删除记录
28                  sql="delect from ORDERREPORT where BookNO="+bno;
                                                              //删除书的 SQL 语句
29                  db.updateDb(sql);                         //执行更新
30              }}
31          catch(Exception e){e.printStackTrace();}          //捕获异常并打印
32      sql="select  BookNo from EXCEEDTIME where BookNO="+bno;
                                                  //检查超期表中是否有该书,若有则删除
33          db.selectDb(sql);                                 //执行检索
34          try{
35              while(db.rs.next()){                          //从超期表中删除记录
36                  sql="delect from EXCEEDTIME where BookNO="+bno;
                                                              //删除记录的 SQL 语句
37                  db.updateDb(sql);                         //执行更新操作
38              }}
39          catch(Exception e){e.printStackTrace();}          //打印异常
40          sql="delete from RECORD where BookNO="+bno;
                                                  //从借书表中将丢失图书的记录删除
41          db.updateDb(sql);                                 //执行更新操作
42          sql="delete from BooK where BookNo="+bno;         //从图书表中将丢失的书删除
43          int i=db.updateDb(sql);                           //执行更新操作
44          db.dbClose();                                     //关闭数据库连接
45          if(i>0){                                          //提示挂失成功
46              JOptionPane.showMessageDialog(this,"恭喜你，挂失成功!!!","消息",
47                              JOptionPane.INFORMATION_MESSAGE);//弹出对话框
48              return;                                       //返回
49          }
50          else{                                             //提示挂失失败
51              JOptionPane.showMessageDialog(this,"对不起，挂失失败!!!",
52                          "信息",JOptionPane.INFORMATION_MESSAGE);//弹出对话框
53              return;                                       //返回
54          }}
```

- 第 4 行代码得到所丢书的书号，第 5 行得到丢书学生的学号，第 6～13 行是在 loseBook 表中搜索所丢书的书名。
- 第 15～20 行是得到丢失书的最大数并加一，作为插入下一条记录的主键，以便向丢失图书表中添加记录。
- 第 22 行、第 23 行是把丢书信息插入 losebook 表中，同时将已经丢失的图书在其他表中的所有信息删除。
- 第 45～54 行是一些提示语句，当挂失成功时，将弹出“挂失成功”提示框，挂失失败将弹出“挂失失败”提示框。

28.9.5　其他方法介绍

28.9.4 节对 loseBook 方法的开发进行了介绍，本节将对图书归还和挂失模块的另外两个方法进行介绍，其详细代码如下。

1. updateTable 方法的开发

这里将介绍在 actionPerformed 方法中曾经调用过的 updateTable 方法的开发，其具体代码如下。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

代码位置：见光盘中本章源代码的 ReturnBook.java 中的 updateTable 方法。

```
1   public void updateTable(){                                    //实现界面下部表格的更新
2       sql="select * from RECORD where StuNO="+jtxt.getText().trim();
                                                                  //检索记录的 SQL 语句
3       db=new DataBase();                                        //创建 DataBase 对象
4       db.selectDb(sql);                                         //将图书信息从数据库中取出
5       Vector<Vector> vtemp = new Vector<Vector>();              //创建向量
6       try{
7           while(db.rs.next()){                                  //遍历结果集
8               Vector<String> v = new Vector<String>();          //创建临时向量
9               for(int i=1;i<7;i++){                             //将每列添加到临时向量 v
10                  String str=db.rs.getString(i);                //得到结果集中的第 i 条记录
11                  str=new String(str.getBytes("ISO-8859-1"),"gb2312");//转码
12                  v.add(str);                                   //添加到临时向量中
13              }
14              vtemp.add(v);                                     //将各条记录添加到 vtemp
15          }
16          db.dbClose();                                         //关闭数据库连接
17      }
18      catch(Exception ea){ea.printStackTrace();}                //捕获异常并打印
19      jt.clearSelection();                                      //删除所选择的行
20      dtm.setDataVector(vtemp,head);                            //更新表格数据
21      jt.updateUI();jt.repaint();                               //更新表格
22  }
```

说明：上一段代码主要是针对表格的更新而开发的，当用户第一次查询的记录被更新以后，可以再次将更新后的记录显示在页面下部的表格中。

2. checkTime 方法的开发

上面介绍了 updateTable 方法，在这里将对图书的超期问题进行处理，其实现代码如下。

代码位置：见光盘中本章源代码的 ReturnBook.java 类中的 checkTime 方法。

```
1   public int checkTime(int sno,int bno){
2       int day=0;                                                //用于记录超期天数
3       int flag=0;                                               //标志位
4       String bname="";                                          //记录书名
5       Date now=new Date();                                      //表示当前系统时间的日期对象
6       String returntime="";                                     //记录归还时间
7       sql="select ReturnTime from RECORD where StuNO="+sno+" and BookNO="+bno;
                                                                  //搜索归还时间
8       db=new DataBase();                                        //从借书信息表中查询某人借某书的归还日期
9       db.selectDb(sql);                                         //执行查询
10      try{
11          if(db.rs.next()){                                     //得到结果集记录
12              returntime=db.rs.getString(1);                    //获取归还时间
13          }}
14      catch(Exception e){e.printStackTrace();}                  //捕获异常并打印
15      String[] strday=returntime.split("\\.");                  //这里使用正则式验证了时间的格式
16      int ryear=Integer.parseInt(strday[0].trim());             //得到当前的年份
17      int rmonth=Integer.parseInt(strday[1].trim());            //得到当前月份
18      int rday=Integer.parseInt(strday[2].trim());              //得到当前天
19      day=(now.getYear()+1900-ryear)*365+(now.getMonth()+1-rmonth)*30+(now.
        getDate()-rday);                                          //超期天
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
20      if(day==-30){                                    //表示当天借的书
21          JOptionPane.showMessageDialog(this,"今天借的书不能还!!! ",
22                                  "消息",JOptionPane.INFORMATION_MESSAGE);//弹出对话框
23          flag=0;                                      //当天借的书则标志位为 0
24      }
25      else if(day>0){                                  //代表已超期
26          int i=JOptionPane.showConfirmDialog(this,"该书已过期,应交罚款为"
27            +day*0.1+"元,是否缴纳罚款?","消息",JOptionPane.YES_NO_OPTION);
                                                         //弹出选择对话框
28          if(i==JOptionPane.YES_OPTION){               //单击“是”时
29              JOptionPane.showMessageDialog(this,"你已成功缴费"+day*0.1+"元",
30                      "消息",JOptionPane.INFORMATION_MESSAGE);//弹出提示对话框
31              flag=-2;                                 //缴钱后标志位为-2
32          }
33          else{                                        //超期且没缴费
34              flag=-1;                                 //欠费，则标志位为-1
35              sql="select BookName from BOOK where BookNO="+bno;
                                                         //搜索书名的 SQL
36              db.selectDb(sql);                        //搜索超期书的书名
37              try{
38                  if(db.rs.next()){bname=db.rs.getString(1).trim();}//得到书名
39              }
40              catch(Exception e){e.printStackTrace();}    //捕获异常并打印
41              sql="insert into EXCEEDTIME(StuNO,BookNO,BookName, DelayTime)
42                       values("+sno+","+bno+",'"+bname+"',"+day+")";
                                                         //插入记录的 SQL
43             db.updateDb(sql);                         //执行更新
44         }}
45      else {                                           //表示可以正常归还的书
46          flag=1;                                      //没有超期，则标示位为 1
47      }
48      db.dbClose();                                    //关闭数据库连接
49      return flag;                                     //返回标志位
50  }
```



说明：上面代码中 flag 的值为“0”，表示该书当天刚借不能归还；“-2”表示该书超期且缴纳了罚金，恢复该用户的借书权限；“-1”表示图书超期但未缴纳罚金，则更新该用户的借书权限为“否”；“1”表示正常归还图书。

至此，图书归还和挂失模块的开发已经基本完成，可以运行 ReturnBook 类中的 main()方法进行测试。



28.10　图书超期模块

28.9 节介绍了归还挂失图书模块的开发，本节将对图书超期模块进行开发，其中包括查询学生用户欠款数，以及交纳欠款功能的完成。在此将分几小节对本模块的开发进行介绍，超期处理模块界面如图 28-20 所示。

28.10.1　界面开发

如果学生所借的图书超期，应该缴纳罚款，首先开发出缴纳罚款的界面，具体代码如下。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

代码位置：见光盘中本章源代码的 ExceedTime.java 类。

```
1   public class ExceedTime extends JFrame implements ActionListener{
2       ……//此处省略了界面的搭建代码
3       public void actionPerformed(ActionEvent e){}//事件处理方法
4   }
```

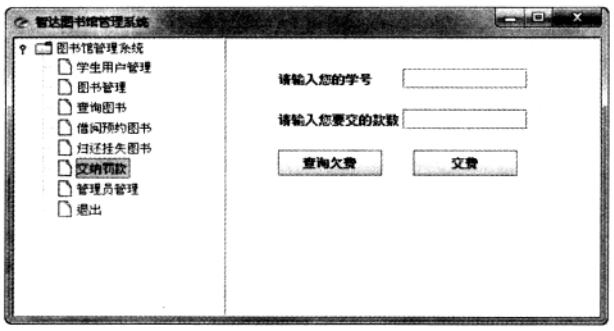


图 28-20　超期处理界面



说明：图书超期模块界面的开发与前面几个模块界面的开发类似。本书由于篇幅所限，在此不再进行详细介绍，读者可以参阅光盘中本章源代码的 ExceedTime.java 类。

28.10.2　查询超期天数

28.10.1 节简单介绍了超期图书模块界面的开发，本节将介绍超期图书信息查询功能的开发，详细代码如下。

代码位置：见光盘中本章源代码的 ExceedTime.java 类中 actionPerformed 方法。

```
1   int day=0;                                          //初始化天数变量
2   DataBase db=new DataBase();                         //创建 DataBase 类对象
3   String sno=(String)jtf.getText().trim();            //定义文本框内容变量
4   if(sno.equals("")){                                 //如果输入学号为空，则进行提示
5       JOptionPane.showMessageDialog(this,"学号不能为空!!!",
6                   "信息",JOptionPane.INFORMATION_MESSAGE);      //弹出信息对话框
7       return;                                         //返回
8   }
9   if(sno.matches("\\D")){                             //定义学号格式为数字组合
10      JOptionPane.showMessageDialog(this,"学号只能为数字!!!",
11                  "信息",JOptionPane.INFORMATION_MESSAGE);   //弹出信息对话框
12      return;                                         //返回
13  }
14  String sql="select DelayTime from EXCEEDTIME where StuNO="+Integer.parseInt
    (sno);                                              //超期天数
15  db.selectDb(sql);                                   //执行查询
16  try{
17      int flag=0;                                     //标志位
18      while(db.rs.next()){                            //得到所借书的超期天数
19          flag++;                                     //标志位自加
20          day+=db.rs.getInt(1);                       //得到超期的天数
21      }
22      if(flag==0){                                    //结果集为空则给出提示
23          JOptionPane.showMessageDialog(this,"您所借的书没有超期，不需要还款!! ",
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



```
24                              "信息",JOptionPane.INFORMATION_MESSAGE);//弹出对话框
25          return;                                          //返回
26      }}
27  catch(Exception ex){ex.printStackTrace();}          //捕获异常并打印
```

- 第 1 行定义的 day 变量用来记录借书天数，初始值为 0。
- 第 4～8 行代码是对输入文本框是否为空进行的处理，为空则弹出提示对话框。
- 第 9～13 行是对输入学号格式进行的处理，这里应用了简单的正则式，验证学号格式是否为纯数字的组合。
- 第 14 行、第 15 行的 SQL 语句是对超期天数进行查询，第 17～26 行代码是执行查询后从结果集中得到超期的天数。

28.10.3　查询及交纳欠款

28.10.2 节实现了对学生用户所借图书超期信息的查询，本节将对超期欠款问题进行解决，详细代码如下。

代码位置：见光盘中本章源代码的 ExceedTime.java 中的 actionPerformed 方法。

```
1   if(e.getSource()==jb1){                                  //事件源为“查询欠款”按钮
2           if(day>0){                                       //提示欠款数
3               JOptionPane.showMessageDialog(this,"您欠费"+day*0.1+"元！",
4                                   "信息",JOptionPane.INFORMATION_MESSAGE);
                                                             //弹出对话框
5           return;                                          //返回
6           }
7           else{                                            //如果没有欠款，则提示
8           JOptionPane.showMessageDialog(this,"您所借的书没有超期，不需要还款!! ",
9                               "信息",JOptionPane.INFORMATION_MESSAGE); //弹出对话框
10          return;                                          //返回
11          }}
12  else if(e.getSource()==jb){                              //事件源为“缴费”按钮
13      if(jtf1.getText().trim().equals("")){                //缴费金额为空的提示
14          JOptionPane.showMessageDialog(this,"请输入缴款金额!! ",
15                              "消息",JOptionPane.INFORMATION_MESSAGE);//弹出对话框
16          return;                                          //返回
17      }
18      int k=JOptionPane.showConfirmDialog(this,"是否缴费?",
19                              "消息",JOptionPane.YES_NO_OPTION);  //弹出选择对话框
20      if(k==JOptionPane.YES_OPTION){                       //当选择“是”时
21          int ii=Integer.parseInt(jtf1.getText().trim());  //得到所交金额
22          if(ii<(day*0.1)){                                //当所交金额小于欠款时
23         sql="update exceedtime set delaytime=delaytime-"+ii/0.1+
24                              " where stuNO="+Integer.parseInt(sno);
                                                             //更新欠款的 SQL 语句
25          db=new DataBase();                               //创建 DataBase 类对象
26          int i=db.updateDb(sql);                          //更新表
27          if(i==1){                                        //输出还应缴费数的提示
28      JOptionPane.showMessageDialog(this,"你已成功缴费"+ii+"元,您还需缴纳"+
29                          (day*0.1-ii)+"元","消息",JOptionPane.INFORMATION_
                            MESSAGE);                        //弹出提示框
30          return;                                          //返回
31          }
32          else{                                            //选择“否”，提示缴费失败
33              JOptionPane.showMessageDialog(this,"对不起，缴费失败!!!",
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
34                                  "消息",JOptionPane.INFORMATION_MESSAGE);
                                                          //弹出提示框
35                  return;                               //返回
36              }}
37      else{                                             //提示缴费成功
38          JOptionPane.showMessageDialog(this,"你已成功缴费"+day*0.1+"元",
39                                  "消息",JOptionPane.INFORMATION_MESSAGE);
                                                          //弹出对话框
40          jtf.setText("");                              //设置文本框内容为空
41          sql="delete from EXCEEDTIME where StuNO="+Integer.parseInt(sno);
                                                          //删除超期记录的 SQL 语句
42          db.updateDb(sql);                             //缴纳了所有欠款更新数据库
43          sql="update STUDENT set Permitted='是' where StuNO="+Integer.
            parseInt(sno);                                //更新权限 SQL 语句
44          db.updateDb(sql);                             //更新 student 表的借书权限
45      } }}
46  db.dbClose();                                         //关闭数据库连接
```

- 第 1～11 行是对查询学生用户欠款金额的实现，这里对 exceedtime 表中的超期天数(delaytime)项信息进行了操作。用 delaytime*0.1 作为计算欠款的公式。
- 第 12～46 行是对“交纳欠款”功能的实现，其中有三种情况：第一种是不缴纳欠款而只查询欠款金额，第二种是缴纳部分欠款，第三种是缴纳全部欠款。在上面的代码中对这三种欠款分别进行了实现，因其较为简单，请读者自行阅读。

至此，图书超期处理模块的开发已经基本完成，可以运行 ExceedTime 类中的 main()方法进行测试。

28.11 学生管理模块

28.10 节已经完成了图书超期处理模块的开发，下面将介绍学生用户的管理功能开发，在此模块开发完毕后，管理员就可以对学生的信息进行管理，学生管理界面效果如图 28-21 所示。

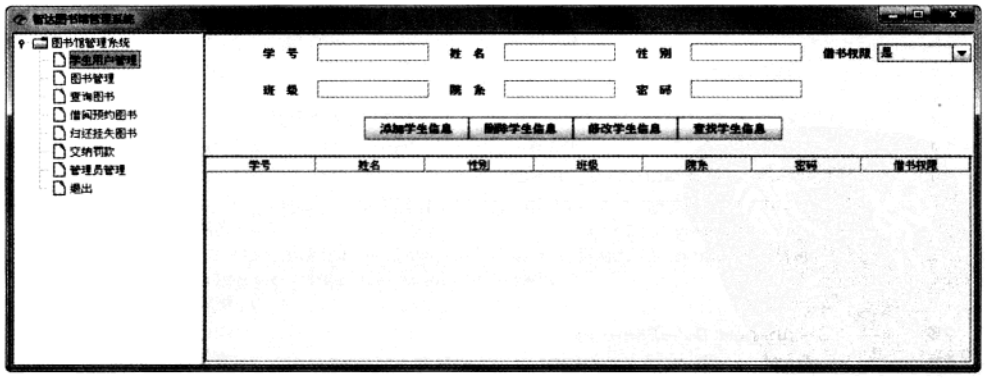


图 28-21 学生用户管理界面

28.11.1 界面开发

学生用户管理主要实现对学生用户的增、删、改、查的各项功能，这里将简单介绍其界面的搭建，具体代码如下。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



代码位置：见光盘中本章源代码的 Student.java 类。

```
1   public class Student extends JFrame implements ActionListener{
2      ……                              //此处省略了变量的声明
3      public Student (){
4         ……                  //此处省略了构造器里面的初始化代码
5      }
6      ……                     //此处省略了一些业务方法
7   }
```



说明：学生用户管理模块界面的开发与学生管理系统几个功能模块界面的开发十分相似，上面几行代码就是对本功能模块界面开发的简单介绍。本书由于篇幅所限，在此不再进行详细介绍，读者可以参阅本书配套光盘中本章源代码 Student.java 类。

28.11.2　响应管理事件

28.11.1 节实现了学生管理界面的搭建，本节将对 Student 类中的 actionPerformed 方法进行介绍，其代码如下。

代码位置：见光盘中本章源代码的 Student.java 类中的 actionPerformed 方法。

```
1       public void actionPerformed(ActionEvent e)
2       {//文本框事件的监听
3            if(e.getSource()==jtxtArray[0]){jtxtArray[1].requestFocus();}
                                          //jtxtArray[1]获得输入焦点
4           if(e.getSource()==jtxtArray[1]){jtxtArray[2].requestFocus();}
                                          //jtxtArray[2]获得输入焦点
5           if(e.getSource()==jtxtArray[2]){jtxtArray[3].requestFocus();}
                                          //jtxtArray[3]获得输入焦点
6            if(e.getSource()==jtxtArray[3]){jtxtArray[4].requestFocus();}
                                          //jtxtArray[4]获得输入焦点
7       if(e.getSource()==jtxtArray[4]){jtxtArray[5].requestFocus();}
                                          //jtxtArray[5]获得输入焦点
8            //当单击“添加学生信息”按钮时将执行添加功能
9            if(e.getSource()==jbArray[0]){this.insertStudent();}
10           //当单击“删除学生信息”按钮时将执行删除功能
11           if(e.getSource()==jbArray[1]){this.deleteStudent();}
12           //当单击“修改学生信息”按钮时将执行修改功能
13           if(e.getSource()==jbArray[2]){this.updateStudent();}
14           //当单击“查找学生信息”按钮时将执行查找功能
15           if(e.getSource()==jbArray[3]){this.searchStudent();}
16      }
```

- 第 3～7 行是监听文本框控件的代码，实现了键盘易用性。关于键盘易用性的功能介绍，请读者参阅本章 28.5.2 节的内容。
- 第 8～15 行是四个功能按钮的事件响应，详细讲解请看代码注释。

28.11.3　操作学生信息

28.11.2 节对 Student 类中的 actionPerformed 方法，以及功能按钮的事件响应进行了介绍。本节将具体介绍对学生用户信息的添加、删除、修改、查询功能的开发，具体代码如下。

代码位置：见光盘中本章源代码的 Student.java 类。

```
1   public void insertStudent(){
2      for(int i=0;i<6;i++){                                    //声明文本框输入信息
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
3            str1[i]=jtxtArray[i].getText().trim();          //得到文本框中的值
4        }
5        if(str1[0].equals("")&&str1[1].equals("")&&str1[2].equals("")
6          &&str1[3].equals("")&&str1[4].equals("")&&str1[5].equals(""))
7        {//当各文本框为空提示
8            JOptionPane.showMessageDialog(this,"学生信息不能为空!!! ",
9                        "消息",JOptionPane.INFORMATION_MESSAGE);//弹出提示对话框
10           return;                                                  //返回
11       }
12       if(!str1[0].equals("")&&!str1[1].equals("")&&!str1[2].equals("")
13         &&!str1[3].equals("")&&!str1[4].equals("")&&!str1[5].equals(""))
14       {//当在文本框输入信息
15           str1[6]=jcp.getSelectedItem().toString();           //得到所选择的值
16           sql="insert into STUDENT(StuNO,StuName,StuSex,Class,Department,"
17               +"Password,Permitted) values('"+str1[0]+"','"+str1[1]+"','"
18               + str1[2] + "',' "+str1[3]+"','"+str1[4]+"','"+str1[5]+"','"+ s
                 tr1[6]+"')";                                    //插入学生的 SQL
19           db=new DataBase();                                  //创建 DataBase 类对象
20           db.updateDb(sql);                                   //插入学生信息
21           Vector<String> v = new Vector<String>();            //创建临时向量
22          for(int i=0;i<=6;i++){                              //将每列添加到临时向量 v
23               v.add(str1[i]);                                //将结果添加到向量中
24               if(i<6){jtxtArray[i].setText("");}             //将文本框设置为空
25          }
26          data.add(v);                                        //将向量 v 添加到表格数据向量中
27           dtm.setDataVector(data,head);                      //设置表格数据
28           jt.updateUI();jt.repaint();                        //执行更新
29           return;                                            //返回
30       }}
31   public void deleteStudent(){
32       String stuno = jtxtArray[0].getText().trim();
33       if(stuno.equals("")){                                   //当学号输入为空时则提示
34           JOptionPane.showMessageDialog(this,"学号不能为空!!! ",
35                       "消息",JOptionPane.INFORMATION_MESSAGE);//弹出对话框
36           return;                                                  //返回
37       }
38       sql="select * from STUDENT where StuNO="+Integer.parseInt(stuno);
                                                                 //搜索学生信息的 SQL
39       db=new DataBase();                                      //创建 DataBase 类对象
40       db.selectDb(sql);                                       //执行查询
41       try{                                                    //对结果集进行异常处理
42           if(db.rs.next()){
43           sql="delete from STUDENT where StuNO="+Integer.parseInt(stuno);
                                                                 //删除学生的 SQL
44           db=new DataBase();                                  //创建 DataBase 类对象
45           db.updateDb(sql);                                   //执行更新
46           JOptionPane.showMessageDialog(this,"成功删除该生信息记录!! ",
47                                       "消息",JOptionPane.INFORMATION_MESSAGE);
                                                                 //弹出对话框
48           return;                                             //返回
49           }
50           else{                                  //若 Record 表中有该学号，则输出提示对话框
51           JOptionPane.showMessageDialog(this,"不能删除该生信息他有书未还,请他还书!!",
52                                       "消息",JOptionPane.INFORMATION_MESSAGE);
                                                                 //弹出对话框
53           }}
54      catch(Exception e){e.printStackTrace();}      //捕获异常并打印
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



```
55      sql="delete from STUDENT where StuNO="+Integer.parseInt(stuno);
                                                    //删除学生信息的 SQL
56      db.updateDb(sql);                           //执行更新
57  }
58  public void updateStudent(){
59      String str[]=new String[7];                          //创建字符串数组
60      int row = jt.getSelectedRow();                       //声明所选行号
61      if(row>=0){                                          //选择了表格中的某行
62          for(int i=0;i<7;i++){str[i]=jt.getValueAt(row,i).toString();}
                                                             //给数组赋值
63          sql="update STUDENT set StuName='"+str[1]+"',StuSex='"+str[2]+"',
64          Class='" +str[3]+"',Department='"+str[4]+"',Permitted='"+str[5]+"',
65          Password='"+str[6]+"' where StuNO="+Integer.parseInt(str[0].trim());
                                                    //更新学生表记录
66          db=new DataBase();                      //创建 DagaBase 类对象
67          db.updateDb(sql);                       //更新学生信息
68          JOptionPane.showMessageDialog(this,"修改成功!! ",
69                      "消息!!",JOptionPane.INFORMATION_MESSAGE);//弹出对话框
70          return;                                                  //返回
71      }
72      if(row==-1){                          //当没有选择就单击“修改信息”按钮 ，则提示
73          JOptionPane.showMessageDialog(this,"请单击'查找'按钮,在下部更改,再选中
74          所改行,"+"单击'修改信息'按钮","Warning!!",JOptionPane.INFORMATION_
            MESSAGE);                               //弹出对话框
75          return;                                 //返回
76      }}
77  public void searchStudent(){
78      if(jtxtArray[0].getText().equals("")){           //文本框内容为空时
79          JOptionPane.showMessageDialog(this,"输入不能为空，请重新输入!!! ",
80                      "信息",JOptionPane.INFORMATION_MESSAGE);//弹出对话框
81          return;                                 //返回
82      }
83      sql="select * from STUDENT where StuNO="+Integer.parseInt(jtxtArray[0].
        getText().trim());
84     db=new DataBase();                       //创建 DataBase 类对象
85      db.selectDb(sql);                       //执行查询
86      try{                                    //对结果集进行异常处理
87          int k=0;                            //标志位，用于判断是否存在该学生
88           Vector<Vector> vtemp = new Vector<Vector>();    //创建向量对象
89           while(db.rs.next()){               //遍历结果集
90              k++;                            //标志位自加
91              Vector<String> v = new Vector<String>();    //创建临时向量对象
92              for(int i=1;i<=7;i++){          //顺序达到所搜到的结果中的各项记录
93                  String str = db.rs.getString(i).trim();//得到结果集中的数据
94                  str = new String(str.getBytes("ISO-8859-1"),"gb2312");
                                                //将 ISO-8859-1 转为 gb2312
95                  v.add(str);                 //向向量中添加数据
96              }
97              vtemp.add(v);                   //更新结果框中的内容
98           }
99           if(k==0){                          //若 Book 表中没有该书号
100              JOptionPane.showMessageDialog(this,    "没有您要查找的内容",
101                "消息",JOptionPane.INFORMATION_MESSAGE);  //弹出提示对话框
102          }
103          dtm.setDataVector(vtemp,head);                   //设置表格数据
104          jt.updateUI();                                   //更新表格
105          jt.repaint();                                    //重绘表格
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
106        }
107        catch(Exception e){e.printStackTrace();}              //捕获异常并打印
108  }
```

- 第 1～30 行是实现学生信息添加的模块。这部分学生信息从界面的文本框中输入，然后显示在下面的表格中，并添加进 student 信息表中。
- 第 31～57 行实现了对学生信息的删除。当某学生因为某种原因离开学校或者已经毕业，其借书的权限已经终止，需要将该生的所有信息从该系统中删除。
- 第 58～76 行实现了学生信息的修改功能。当管理员得知某学生的信息有错误，就需要进行修改，首先从数据库中将该生的所有信息查询出来，并显示在表格中，需要修改哪项，就可以很方便地对其进行修改。
- 第 77～108 行实现了对学生信息的查询。当需要了解某学生用户的信息时，就可以对其进行查询并显示在表格中。

至此，学生管理模块的开发已经基本完成，可以运行 Student 类中的 main()方法进行测试。

28.12 图书管理模块

对于图书馆来说，最需要进行管理的资源是图书，图书信息的管理占管理员工作的很大一部分，下面将对图书管理模块的开发进行介绍，图书管理模块界面如图 28-22 所示。

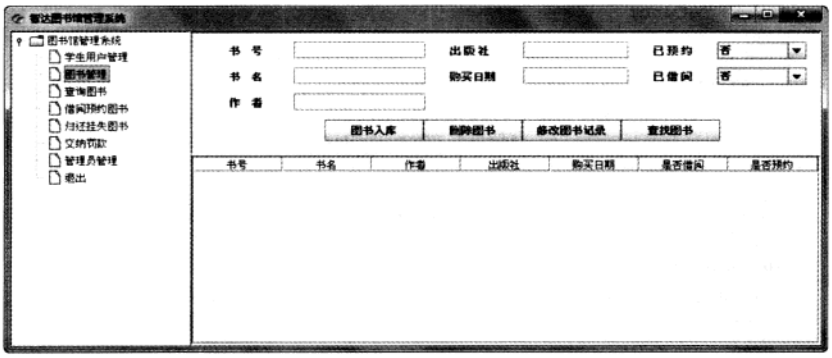


图 28-22 图书管理界面

28.12.1 界面开发

图书的管理主要包括图书入库、删除图书记录、修改图书信息和查询图书等操作，本节首先简单介绍其管理界面的搭建，具体代码如下。

代码位置：见光盘中本章源代码的 BookManage.java

```
1   public class BookManage extends JFrame implements ActionListener{
2       ……//省略了界面变量的声明
3       public BookManage(){
4        ……//省略了构造器代码
5       }}
```

说明：由于本模块的界面与上一模块的界面类似，具体的设计方法请参考前几节。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



28.12.2　响应管理事件

开发完界面后，就可以开发对界面中控件的监听代码，即实现 ActionListener 接口中的 actionPerformed 方法，具体代码如下。

代码位置：见光盘中本章源代码的 BookManage.java

```
1   public void actionPerformed(ActionEvent e){
2       if(e.getSource()==jtxtArray[0]){jtxtArray[1].requestFocus();}
                                                          //设置输入焦点
3       if(e.getSource()==jtxtArray[1]){jtxtArray[2].requestFocus(); }
                                                          //设置输入焦点
4       if(e.getSource()==jtxtArray[2]){jtxtArray[3].requestFocus(); }
                                                          //设置输入焦点
5       if(e.getSource()==jtxtArray[3]){jtxtArray[4].requestFocus(); }
                                                          //设置输入焦点
6       if(e.getSource()==jbArray[0]){this.insertBook();}    //添加图书
7      if(e.getSource()==jbArray[1]){this.deleteBook();}
                                                  //将书号为书号框中的书从书库删除
8       if(e.getSource()==jbArray[2]){this.updateBook();}
                                                  //对书号为书号框中的书信息进行修改
9       if(e.getSource()==jbArray[3]){this.searchBook();}    //查询图书信息
10  }
```



说明：以上代码中实现了键盘易用性，这对进行大量操作的图书管理员来说，工作量大大减少，因此，提高了管理员的工作效率。

28.12.3　操作图书信息

管理员可以通过单击“添加图书”、“删除图书”、“查询图书”按钮对图书的信息进行增加、删除和查询操作。下面将详细讲解这几个功能的开发，其代码如下。

代码位置：见光盘中本章源代码的 BookManage.java 类。

```
1   public void insertBook(){
2       for(int i=0;i<5;i++){                                  //声明输入变量
3         str1[i]=jtxtArray[i].getText().trim();               //得到文本框内容
4       }
5       if(str1[0].equals("")&&str1[1].equals("")&&str1[2].equals("")
6             &&str1[3].equals("")&&str1[4].equals("")){       //当输入为空则提示
7           JOptionPane.showMessageDialog(this,     "图书信息不能为空!!! ",
8               "消息",JOptionPane.INFORMATION_MESSAGE);       //弹出信息提示框
9         return;                                              //返回
10      }
11     if(!str1[0].equals("")&&!str1[1].equals("")&&!str1[2].equals("")
12          &&!str1[3].equals("")&&!str1[4].equals("")){//将图书信息插入 Book 表
13          str1[5]=jcp1.getSelectedItem().toString();  //得到所选择的值
14          str1[6]=jcp2.getSelectedItem().toString();  //得到所选择的值
15          sql="insert into BOOK values('"+str1[0]+"','"+str1[1]+"','"
16           + str1[2] + "',' "+str1[3]+"','"+str1[4]+"','"+str1[5]+"',
            '"+str1[6]+"')";                                  //插入 BOOK 表的 SQL
17          db=new DataBase();                                //创建 DataBase 类对象
18          db.updateDb(sql);                                 //数据插入
19          Vector<String> v = new Vector<String>();          //创建临时向量
20        for(int i=1;i<=7;i++){                              //将每列添加到临时向量 v
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
21                v.add(str1[i-1]);                    //将数组中的信息添加到向量中
22            }
23            data.add(v);                             //将临时向量添加到 data 中
24             dtm.setDataVector(data,head);           //设置表格数据
25             jt.updateUI();jt.repaint();                  //更新 table
26             return;                                      //返回
27        }}
28    public void deleteBook(){
29        String bookno = jtxtArray[0].getText().trim();  //得到书号
30        if(bookno.equals("")){                           //当书号输入为空，则提示
31            JOptionPane.showMessageDialog(this,     "书号不能为空!!! ",
32                          "消息",JOptionPane.INFORMATION_MESSAGE);//弹出对话框
33            return;                                      //返回
34        }
35        sql="select * from RECORD where BookNO="+Integer.parseInt(bookno);
                                                           //搜索记录的 SQL
36        db=new DataBase();                               //创建 DataBase 类对象
37        db.selectDb(sql);                                //删除图书记录
38        try{
39            if(db.rs.next()){                            //若 Record 表中有该书号
40                JOptionPane.showMessageDialog(this,"不能删除此书",
41                                  "消息",JOptionPane.INFORMATION_MESSAGE);
                                                           //弹出对话框
42                return;                                  //返回
43            }}
44       catch(Exception e){e.printStackTrace();}          //捕获异常并打印
45        sql="delete from book where BookNO="+Integer.parseInt(bookno);
                                                           //删除 book 表中记录
46        db.updateDb(sql);                                //执行更新
47    }
48    public void searchBook(){                            //弹出提示对话框
49        JOptionPane.showMessageDialog(this,"请单击左边窗口的 '查询图书' 按钮!! ",
50                                  "强烈推荐",JOptionPane.INFORMATION_MESSAGE);
                                                           //弹出对话框
51       return;                                           //返回
52    }
```

- 第 1～27 行代码是实现“添加图书”功能的模块，其对应“添加图书”按钮，当单击“添加图书”按钮时，将执行这段代码。
- 第 28～47 行代码实现“添加图书”功能，其对应“删除图书”按钮，当单击“删除图书”按钮时将执行这段代码。
- 第 48～52 行代码是对应“查询图书”按钮的代码，因为此按钮功能与系统中子节点“查询图书”的功能冲突，故在此提示到节点“查询图书”界面进行查询。

28.12.4 修改图书信息

28.12.3 节介绍了对图书信息的增加、删除和查询功能的开发，本节将介绍对图书信息修改功能的开发，其代码如下。

代码位置：见光盘中本章源代码的 BookManage.java 类中的 updateBook 方法。

```
1    public void updateBook(){
2        String bookno = jtxtArray[0].getText().trim();  //得到书号
3        if(bookno.equals("")){                           //当书号输入为空时，进行提示
4            JOptionPane.showMessageDialog(this,"请输入需要更改信息图书的书号! ",
5                          "消息",JOptionPane.INFORMATION_MESSAGE); //弹出对话框
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
6            return;                                    //返回
7        }
8        else{                                          //当输入书号后的情况
9           for(int i=0;i<5;i++){                       //得到文本框输入的图书信息
10            str1[i]=jtxtArray[i].getText().trim();     //得到文本框的内容
11          }
12           db=new DataBase();                         //创建数据库类对象
13           int i=0;                                   //用于记录哪个没有填写
14           int flag=0;                                //记录更新条数
15           int b=Integer.parseInt(bookno);            //将书号转成整型数据
16           if(!str1[1].equals("")){i=i+1;}            //str[1]为空时
17           if(!str1[2].equals("")){i=i+2;}            //str[2]为空时
18           if(!str1[3].equals("")){i=i+4;}            //str[3]为空时
19           if(!str1[4].equals("")){i=i+8;}            //str[4]为空时
20           switch(i){                                 //请详细解读 switch 语句
21                case 0:
22                    JOptionPane.showMessageDialog(this,"信息不能为空!!! ",
23                          "消息",JOptionPane.INFORMATION_MESSAGE);//弹出对话框
24                break;                                //退出 switch 语句块
25              ……                                    //省略了 case 1~ case 14 的代码
26              case 15:
27              sql="update BOOK set BookName='"+str1[1]+"',"+"Author='"+str1[2]+
28               "',"+"Publishment='"+str1[3]+"',"+"BuyTime='"+str1[4]+"'   where
                 BookNO="+b;
29              flag=db.updateDb(sql);                 //更新书名、作者、出版社和购进时间信息
30              if(flag>0){                            //更新成功
31                    JOptionPane.showMessageDialog(this,"恭喜你，修改成功!!!",
32                          "消息",JOptionPane.INFORMATION_MESSAGE);//弹出对话框
33              }
34                break;                                //退出 switch 语句块
35           }}}
```

- 第 3～7 行为当用户修改后的书号为空时，弹出消息提示对话框。
- 第 16～19 行是四个 if 语句：当“书名”文本框输入不为空时，将 i 值置为 1；当“作者”文本框输入不为空时，将 i 值置为 2；当“出版社”文本框输入不为空时，将 i 值置为 4；当“购进时间”文本框输入不为空时，将 i 值置为 8。这与用四位二进制数表示十进制数字的道理相同。
- 从第 20～35 行判断 i 值的不同情况，即对用户输入图书信息进行判断，根据不同情况进行不同的处理。

至此，图书管理模块的开发已经基本完成，可以运行 BookManage 类中的 main()方法进行测试。



28.13　管理员管理模块

图书馆管理员是图书馆管理信息系统的管理者，其具有对学生信息和图书信息的管理权限，并需要对图书馆管理系统进行维护，管理员管理模块如图 28-23 所示，下面将对管理员管理模块的开发进行介绍。

28.13.1　界面开发

管理员分为超级管理员和普通管理员，超级管理员可以对普通管理员进行管理，而普通管

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

理员则只能管理图书信息和学生用户信息，下面首先简单介绍界面的搭建，其代码如下。

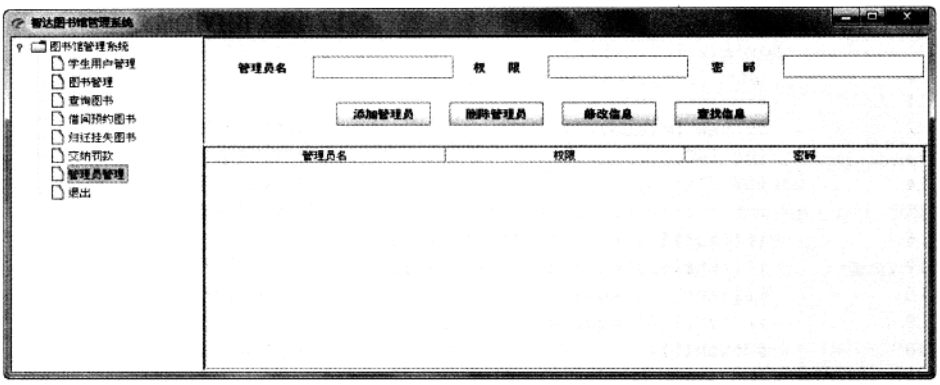


图 28-23 管理员管理界面

代码位置：见光盘中本章源代码的 Manager.java 类。

```
1   public class Manager extends JFrame implements ActionListener{
2     public Manager(String mgNo){
3         ……                                          //省略构造器
4     }
5     public void setFlag(boolean b)
6     {                                                 //设置按钮是否可编辑
7        jbArray[0].setEnabled(b);                      //设置 jbArray[0]的可编辑性
8        jbArray[1].setEnabled(b);                      //设置 jbArray[1]的可编辑性
9      }
10     public void actionPerformed(ActionEvent e){} //实现动作事件处理方法
11  }
```

说明：因本界面与前面其他模块界面的设计类似，故在此不再赘述，详细代码请参阅本章光盘中的内容。

28.13.2 响应管理事件

前面介绍了本模块的界面开发，本节将介绍对管理员管理界面中各个控件的监听代码的开发，具体代码如下。

代码位置：见光盘中本章源代码的 Manager.java 类中的 actionPerformed 方法。

```
1   if(e.getSource()==jtxtArray[0]){jtxtArray[1].requestFocus();}
                                                      //jtxtArray[1]获得输入焦点
2   if(e.getSource()==jtxtArray[1]){jtxtArray[2].requestFocus();}
                                                      //jtxtArray[2]获得输入焦点
3   sql="select permitted from manager where mgNo='"+mgNo+"'";//查询搜索权限的 SQL
4   db=new DataBase();                                //创建数据库操作类对象
5   db.selectDb(sql);                                 //查询管理员权限
6   String string="";                                 //用于记录管理员权限
7   try{
8       while(db.rs.next()){                          //遍历结果集
9           string=db.rs.getString(1).trim();         //得到权限
10      }
11      int p=0;                                      //标志位
12      if(string.equals("1")){p++;                   //当是超级管理员时
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



```
13          String jtxt=jtxtArray[0].getText().trim();       //得到文本框中的值
14          if(jtxt.equals("")){                             //当输入为空，则提示
15                  OptionPane.showMessageDialog(this, "请输入管理员名!!! ",
16                        "消息",JOptionPane.INFORMATION_MESSAGE);//弹出对话框
17              return;                                      //返回
18          }
19          if(e.getSource()==jbArray[0]){                   //执行添加操作
20              this.insertManager();                        //调用插入管理员方法
21          }
22          if(e.getSource()==jbArray[1]){                   //执行删除操作
23              this.deleteManager();                        //调用删除管理员方法
24          }
25          if(e.getSource()==jbArray[2]){                   //执行修改操作
26              this.updateManager();                        //调用更新管理员方法
27          }
28          if(e.getSource()==jbArray[3]){                   //执行查询操作
29              this.selectManager();                        //调用查询管理员方法
30          }}
31      if(p==0){                                            //说明该管理员是普通管理员
32          jtxtArray[0].requestDefaultFocus();
33          String jtxt=jtxtArray[0].getText().trim();
34          if(jtxt.equals("")){                             //管理员名输入为空，提示
35              JOptionPane.showMessageDialog(this,     "请输入管理员名!!! ",
36                              "消息",JOptionPane.INFORMATION_MESSAGE);
                                                             //弹出对话框
37               return;                                     //返回
38          }
39          else if(jtxt.equals(mgNo)){                      //输入了管理员名
40              if(e.getSource()==jbArray[3]){               //执行 manager1 方法
41                  this.manager1();                         //调用 manager1 方法
42              }
43              if(e.getSource()==jbArray[2]){               //执行修改操作
44                  String str[]=new String[3];              //创建字符串数组
45                  int row = jt.getSelectedRow();           //得到所选择的行
46                  if(row>=0){                              //当选中行时
47                      for(int i=0;i<3;i++){
48                          str[i]=jt.getValueAt(row,i).toString();
                                                             //得到所选择的值
49                      }
50                      sql="update manager set password='"
51                          +str[2]+"' where mgNo="+Integer.parseInt(str[0].
                            trim());                         //更新表的 SQL
52                          db=new DataBase();               //创建 DataBase 类的方法
53                          int i=db.updateDb(sql);          //重新设置密码
54                          if(i==1){                        //更新成功
55                              JOptionPane.showMessageDialog(this,"修改成功!!",
56                               "消息!!",JOptionPane.INFORMATION_MESSAGE);
                                                             //弹出对话框
57                              return;                      //返回
58                          }
59                          else{                            //操作错误提示
60                              JOptionPane.showMessageDialog(this,"请重新操
                                作!! ",
61                           "Warning!!",JOptionPane.INFORMATION_MESSAGE);
                                                             //弹出对话框
62                              return;                      //返回
63                          }}
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
64                          else{                                   //操作错误提示
65                              JOptionPane.showMessageDialog(this,"请重新操作!! ",
66                            "Warning!!",JOptionPane.INFORMATION_MESSAGE);
                                                                    //弹出对话框
67                              return;                             //返回
68                          }}}
69                  else{                                           //提示普通管理员没有的权限
70                      JOptionPane.showMessageDialog(this,"你不能查看别人的信息,请重
                        新输入！",
71                        "消息",JOptionPane.INFORMATION_MESSAGE);   //弹出对话框
72                    return;                                       //返回
73                  }}}
74          catch(Exception ex){ex.printStackTrace();}              //捕获异常并打印
75          db.dbClose();                                           //关闭数据库连接
```

- 第 3～10 行是对管理员权限的判断，不同权限的管理员有不同的管理级别，如普通管理员只有修改自己密码的权限，而超级管理员可以对管理员的一切信息进行增、删、改、查。
- 第 12～30 行代码是对超级管理员的权限进行开发。第 19～30 行代码分别实现了其增、删、改、查的权限。详细的功能实现代码将在后面介绍。
- 第 31～75 行代码实现了普通管理员对系统的管理功能及对自己密码的修改功能，其中调用了 manager1 方法将在后面详细介绍。

28.13.3 普通管理员功能

28.13.2 节介绍了用 actionPerformed 方法响应管理事件，本节将介绍在 28.13.2 节中调用过的 manager1 方法，详细代码如下所列。

代码位置：见光盘中本章源代码的 Manager.java 类中的 manager1 方法。

```
1   public void manager1(){
2       sql="select * from manager where mgNo="+Integer.parseInt(jtxtArray[0].
        getText().trim());                                          //搜索的 SQL
3       db=new DataBase();                                          //创建 DataBase 类的对象
4       db.selectDb(sql);                                           //执行查询
5       try{                                                        //进行异常处理
6           int k=0;                                                //记录是否检索到数据
7            Vector<Vector> vtemp = new Vector<Vector>();    //创建向量
8            while(db.rs.next()){                                   //取得结果集
9               k++;                                  //标志位自加
10              Vector<String> v = new Vector<String>();    //创建临时向量
11              for(int i=1;i<=3;i++){                //顺序找到所搜到的结果中的各项记录
12                  String str=db.rs.getString(i).trim();
13                  str=new String(str.getBytes("ISO-8859-1"),"gb2312");//转码
14                  v.add(str);                       //添加到向量中
15              }
16              vtemp.add(v);                         //添加到表格数据向量
17           }
18           if(k==0){                                //若 manager 表中没有该管理员号
19               JOptionPane.showMessageDialog(this,"没有您要查找的内容",
20                          "消息",JOptionPane.INFORMATION_MESSAGE);//弹出对话框
21           }
22          dtm.setDataVector(vtemp,head);        //设置表格数据
23          jt.updateUI();jt.repaint();           //更新表格
24      }
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
25        catch(Exception e){e.printStackTrace();}              //捕获异常并打印
26    }
```

说明：manager1 方法是一个很简单的方法，其主要实现了管理员名是“管理员名”文本框中输入信息的管理员记录的搜索，并将搜索结果显示在表格中的功能。

28.13.4　添加管理员

超级管理员可以添加普通管理员，本节将实现添加管理员的功能，下面对其调用的添加管理员的数据库方法进行介绍，具体代码如下。

代码位置：见光盘中本章源代码的 Manager.java 类。

```
1    public void insertManager(){
2        for(int i=0;i<3;i++){
3            str1[i]=jtxtArray[i].getText().trim();          //得到文本框中的值
4        }
5        if(!str1[0].equals("")&&!str1[1].equals("")&&!str1[2].equals("")){
                                                             //当文本框均不为空时
6            if(!str1[0].matches("^\\d+$")){                 //判断管理员名的格式
7                JOptionPane.showMessageDialog(this,"管理员名只能为数字!!! ",
8                        "消息",JOptionPane.INFORMATION_MESSAGE);//弹出对话框
9                return;                                     //返回
10           }
11           int temp = Integer.parseInt(str1[0]);           //将 str1[0]转换为 int 型
12           sql="insert into manager(mgNo,Permitted,password) values("
13                   +temp+",'"+str1[1]+"','"+str1[2]+"')";
                                                 //得到向 manager 表中插入记录的 SQL
14           db=new DataBase();                  //创建 DataBase 类对象
15           int j=db.updateDb(sql);             //执行更新
16           if(j==0){                           //提示失败信息
17               JOptionPane.showMessageDialog(this,     "插入失败!!! ",
18                       "消息",JOptionPane.INFORMATION_MESSAGE);   //弹出对话框
19               return;                                     //返回
20           }
21           Vector<String> v=new Vector<String>();          //创建临时向量
22           for(int i=0;i<=2;i++){
23               v.add(str1[i]);                             //将每列添加到临时向量 v
24               if(i<3){jtxtArray[i].setText("");}          //设置文本框内容为空
25           }
26           data.add(v);                                    //将临时向量添加到 data 中
27           dtm.setDataVector(data,head);                   //设置表格数据
28           jt.updateUI();jt.repaint();                     //更新表格
29           return;                                         //返回
30       }
31       else{                                               //输入为空时提示
32           JOptionPane.showMessageDialog(this,     "管理员信息不能为空!!! ",
33                   "消息",JOptionPane.INFORMATION_MESSAGE);//弹出对话框
34           return;                                         //返回
35       }}
```

- 第 6 行代码中使用了正则式来规定管理员名的格式只能为数字的组合。
- 第 12 行的 SQL 语句是把输入的信息插入 manager 表中，如果插入失败，将给出提示信息。
- 第 21～29 行是把插入的管理员信息显示在下部的表格中，如果插入为空，将给出提示。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

- 第 31～35 行代码是对管理员信息输入为空的情况进行处理，此时会弹出提示对话框。



说明：当单击“添加管理员”按钮时，就可以把输入到文本框的信息添加到 manager 表里作为一名管理员的一条信息。要注意，超级管理员的权限值设为“1”，普通管理员的权限值设为“0”。

28.13.5 删除管理员功能

28.13.4 节主要介绍添加管理员功能，本节将对删除管理员功能进行开发，具体代码如下。

代码位置：见光盘中本章源代码的 Manager.java 类。

```
1   public void deleteManager(){
2       String mgNo=jtxtArray[0].getText().trim();        //得到管理员号
3       if(mgNo.equals("")){                              //为空则给出提示
4           JOptionPane.showMessageDialog(this,     "学号不能为空!!! ",
5                                   "消息",JOptionPane.INFORMATION_MESSAGE);
                                                          //弹出信息对话框
6           return;                                       //返回
7       }
8       sql="select permitted from manager where mgNo="+Integer.parseInt(mgNo);
                                                          //搜索管理员权限 SQL
9       db=new DataBase();                                //创建 DataBase 类对象
10      db.selectDb(sql);                                 //执行删除
11      String str="";                                    //用于存放管理员权限
12      try{
13          while(db.rs.next()){                          //遍历结果集
14              str=db.rs.getString(1).trim();            //得到结果集中的数据
15          }
16          if(str.equals("1")){                          //输出提示信息
17          JOptionPane.showMessageDialog(this, "不能删除超级管理员信息记录!! ",
18                                  "警告",JOptionPane.INFORMATION_MESSAGE);
19          return;                                       //返回
20          }
21          else{                                         //执行删除
22              sql="delete from manager where mgNo="+Integer.parseInt(mgNo);
                                                          //删除管理员语句
23              db=new DataBase();                        //创建 DataBase 类对象
24              db.updateDb(sql);                         //更新数据库
25              JOptionPane.showMessageDialog(this,"成功删除该管理员信息!! ",
26                  "消息",JOptionPane.INFORMATION_MESSAGE);     //弹出提示框
27          }}
28      catch(Exception e){e.printStackTrace();}          //捕获异常并打印
29  }
```



说明：这个功能只是超级管理员才有的。当某个管理员因为某种原因离开了其工作岗位，这就需要将其所有信息删除，此时，超级管理员只要输入需要删除的管理员名，然后单击“删除管理员”即可。

28.13.6 修改管理员信息

28.13.5 节介绍了超级管理员删除管理员信息功能，本节将对修改管理员信息功能进行开

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

发，其详细代码如下。

代码位置：见光盘中本章源代码的 Manager.java 类。

```
1   public void updateManager(){
2       String str[]=new String[3];                          //创建字符串数组
3       int row = jt.getSelectedRow();                       //得到选择行
4       if(row>=0){                                          //选中行时
5           for(int i=0;i<3;i++){                            //得到所选内容
6               str[i]=jt.getValueAt(row,i).toString();//得到所选择的值
7           }
8           sql="update manager set mgNo='"+str[0]+"',permitted='"+str[1]
9             +"',password='"+str[2]+"' where mgNo="+Integer.parseInt(str[0].
              trim());                                       //更新管理员表的 SQL
10          db=new DataBase();                               //创建 DataBase 类对象
11          db.updateDb(sql);                      //执行修改功能，并给出修改成功提示
12          JOptionPane.showMessageDialog(this,"修改成功!! ",
13              "消息!!",JOptionPane.INFORMATION_MESSAGE);  //弹出对话框
14          return;                                          //返回
15      }
16      if(row==-1){                                         //对操作步骤进行提示
17          JOptionPane.showMessageDialog(this,
18          "请单击'查找'按钮,在下部的表格里更改,再选中所改行,单击'修改信息'按钮！",
19                  "Warning!!",JOptionPane.INFORMATION_MESSAGE);//弹出对话框
20          return;                                          //返回
21      }}
```



说明：这个功能对于不同权限的管理员，其操作是不相同的，超级管理员可以对任意管理员的信息进行修改，而普通管理员只可以修改自己的密码，这样的功能设置对系统的稳定，以及图书馆的财产安全是有一定好处的。

28.13.7　查询管理员信息

前面分别介绍了超级管理员的增、删、改功能的开发，本节将对查询功能进行开发，详细代码如下。

代码位置：见光盘中本章源代码的 Manager.java 类。

```
1       public void selectManager(){
2        if(jtxtArray[0].getText().equals("")){                    //给出提示
3           JOptionPane.showMessageDialog(this,"输入不能为空，请重新输入!!! ",
4                          "信息",JOptionPane.INFORMATION_MESSAGE);//弹出对话框
5           return;                                                //返回
6       }
7           int q=Integer.parseInt(jtxtArray[0].getText().trim());//得到管理员号
8           sql="select * from manager where mgNo="+q;  //搜索管理员信息
9         db=new DataBase();                                       //创建 DataBase 类对象
10          db.selectDb(sql);                                      //查询管理员信息
11          try{
12              int k=0;                                           //用于记录是否搜索出记录
13               Vector<Vector> vtemp = new Vector<Vector>();    //创建向量
14               while(db.rs.next()){                              //遍历结果集
15                  k++;                                           //k 自加
16                  Vector<String> v = new Vector<String>();    //创建临时向量
17                  for(int i=1;i<=3;i++){          //顺序得到所查询结果中的各项记录
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
18                          String str=db.rs.getString(i).trim();//得到结果集中的数据
19                          str=new  String(str.getBytes("ISO-8859-1"),"gb2312");
                                                        //将 ISO-8859-1 转为 gb2312
20                          v.add(str);                 //将结果添加到 v 中
21                      }
22                      vtemp.add(v);                   //将得到的数据添加到表格数据向量
23                  }
24                  if(k==0){                           //若 Book 表中没有该书号
25                       JOptionPane.showMessageDialog(this,    "没有您要查找的内容",
26                            "消息",JOptionPane.INFORMATION_MESSAGE);//弹出对话框
27                  }
28                  dtm.setDataVector(vtemp,head);      //设置表格数据
29                  jt.updateUI(); jt.repaint();        //更新表格
31              }
32              catch(Exception e){e.printStackTrace();}            //捕获异常并打印
33      }
```

说明：当单击“查询信息”按钮时，就会执行查询功能，超级管理员可以查询任意管理员的任意信息，这样有利于对其进行统一管理，提高工作效率。

至此，管理员管理模块的开发已经基本完成，可以运行 Manager 类中的 main()方法进行测试。

28.14 集成和测试

学生端的操作权限仅限于查询图书、预约图书和挂失图书等，而这些功能的开发与管理模块类似。所以，本书就不再介绍学生端的开发，有兴趣的读者可参阅光盘中本章源码自己进行开发。下面对学生端开发的几个类进行简单说明。

- StudentSystem 类：此类是学生端主框架的搭建类，具体和管理端的 Root 类功能类似。
- 查询图书类：由于此功能和管理端一样，所以就重用了管理端的查询图书类 SearchBook。
- OrderBook 类：学生端预约图书模块，和管理端预约借阅图书模块类似，只是少了借阅功能。
- LossReport 类：学生端挂失图书模块，和管理端挂失归还模块类似，只是少了归还图书模块。

提示：学生端的开发主要考虑学生权限问题，考虑学生在图书管理系统中能执行什么样的功能，因此，业务功能需要符合实际情况。

下面将各个模块集成到主框架中，具体步骤如下。

① 将各个模块开发的 SearchBook、BorrowBook、ReturnBook、ExceedTime、Student、BookManage 和 Manager 类改为继承自 JPanel，因为各个模块测试阶段为了方便测试，均继承自 JFrame。

② 做完第①步后重新编译以上类，并编译主框架类 Root。

③ 运行登录窗体类，并输入正确的管理员的用户名和密码，登录进入管理端主窗体界面如图 28-9 所示，在此界面选中右边树状列表的各个节点均可显示相应的管理界面，在各个功能界面均能进行正常功能操作。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



提示：具体的数据测试请读者自行完成，这里不再赘述。



28.15　总结

本章详细介绍了一个具有基本功能的图书管理系统的开发，通过对本章的学习，读者应该对 Swing 控件的应用及布局有了更深的理解。在实际开发中，读者可以根据该图书信息管理系统实例并结合实际情况，开发出功能更加完善的信息管理系统。在开发过程中用到了许多开发技巧及重要知识点，下面将对这些技巧及知识点进行简单介绍。

- Swing 控件的应用：Swing 控件是完全轻量级控件，其跨平台性使这些控件开发的界面在不同的操作系统中显示的感官效果是相同的，不会因平台不同而造成显示的差异。在本系统的开发中使用了面板、标签、按钮、文本框、密码框、单选按钮、分割窗格、下拉列表框等。
- 树状列表的应用：在本系统中，使用了树状列表的结构来进行页面的导航，这样使系统的界面结构清晰、操作简单。因此，在当前比较受客户的欢迎。
- 表格的应用：实际开发中，经常需要在界面中显示表格形式的数据，这时就需要使用表格控件。其主要负责数据的显示，并不负责对数据的管理，具体应用在本系统中有多处，在此就不一一列举了。
- 提高键盘易用性：所谓键盘易用性，一般是指在需要输入很多不同信息的界面中，在一种信息输入完成后，按“Enter”键，输入焦点自动跳转到下一个要输入信息的控件。为了提高工作效率，提高键盘易用性十分必要。在本系统中就实现了这种人性化的功能。
- 正则式的应用：正则式主要是对输入的字符串进行格式匹配，本系统中，主要使用 Matcher 类中的 matches 方法，这个方法只能用来检测整个字符串是否满足匹配，而不能进行部分匹配的检测。
- 卡片布局的应用：系统中管理员端与学生用户端显示各功能模块是通过卡片布局完成的，大家可以体会到其直观、简洁的优点。详细介绍请读者参阅学生管理系统与图书管理系统模块，或者参阅本章源代码中的 StudentSystem.java 类和 Root.java 类。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。